IMPROVING BROADBAND NOISE FILTER FOR AUDIO SIGNALS

A Thesis
presented to
The Faculty of California Polytechnic State University,
San Luis Obispo

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in Electrical Engineering

by
Alexander Conway
May 2012

COMMITTEE MEMBERSHIP

TITLE: Improving Broadband Noise Filter for Audio Signals

AUTHOR: Alexander Conway

DATE SUBMITTED: May 2012

COMMITTEE CHAIR: Wayne Pilkington, Associate Professor

COMMITTEE MEMBER: Fred DePiero, Professor

COMMITTEE MEMBER: John Saghri, Associate Professor

Abstract

Improving broadband noise filter for audio signals

By

Alex Conway

Filtering broadband noise in audio signals has always been a challenge since the noise is spread across a large bandwidth and overlaps the spectral range of the audio signal being recovered. There have been several broadband noise reduction techniques, such as spectral subtraction, developed for speech enhancement that might be applied to reduce noise in musical audio recordings as well. One such technique that is investigated in this thesis identifies the harmonic components of a signal to be preserved as those spectral magnitude peaks that exceed a chosen decision threshold. All other spectral components below the threshold are considered to be noise. Noise components are then attenuated enough to render them imperceptible in the presence of the harmonic signal, using a psycho-acoustical model of sound masking.

The objective of this thesis is to show that this algorithm can be adapted and improved for musical audio noise reduction. Improvements include relaxing the filter when percussion events are anticipated, since these appear spectrally similar to broadband noise but are essential to the musical experience; and using harmonic prediction to preserve more of the signal harmonics than the simple threshold would pass. Harmonic prediction takes advantage of the known harmonic

spacing of musical content and places noise filter pass bands around a fixed

number of harmonics of every fundamental pitch found; passing some harmonics that would get filtered out by the original thresholding algorithm.

Noise filtering improvements were assessed for both noise reduction and signal degradation effects by different signal to noise ratio computations. The audio results with and without improvements were also assessed using subjective listening tests, since how the sound is perceived is what matters most in musical recordings. Quantitative results show improved signal to noise ratios of filtered audio signals when the improvements were included compared to the original threshold-based filter. Perceived sound quality in listening tests was also higher with the percussion preservation and harmonic prediction improvements. In all listening tests, every listener rated the improved filter as the best.

Table of Contents

LIST OF TABLES…….....ix

LIST OF FIGURES……..x

1 INTRODUCTION……...1

2 FUNDAMENTALS…….3

2.1 AUDIO SIGNALS……...3

2.2 DECISION THRESHOLD……..5

2.2.1 Median and Moving Average Filtering……5

2.3 PSYCHOACOUSTICS……...6

2.3.1 Absolute Threshold of Hearing……...6

2.3.2 Critical Band Frequency Analysis……8

2.3.3 Simultaneous Masking……..9

3 EXISTING NOISE SUPPRESSION ALGORITHMS..…….....12

3.1 SPECTRAL SUBTRACTION……..12

3.1.1 Introduction……...12

3.1.2 Spectral Subtraction Theory……12

3.2 WIENER FILTERING……..14

3.2.1 Introduction……..14

3.2.2 Wiener Filter Theory……..15

3.3 DOLBY NOISE REDUCTION…….18

3.3.1 Introduction……..18

3.3.2 Dolby B……18

4 BROADBAND NOISE REDUCTION ALGORITHM……22

4.1 THE BROADBAND NOISE FILTER……..23

4.1.1 Adaptive Threshold Curve……24

4.1.2 Masking Threshold Calculation……..27

4.1.3 Suppression Function……33

4.2 NOISE SUPPRESSION FILTER SOUND FIDELITY IMPROVEMENTS……34

4.2.1 Preservation of Signal Harmonics……..35

4.2.2 Song Tempo Synchronization Processing……..37

4.2.3 Percussion Preservation (drums, cymbals, etc.)……...38

5 TEST & RESULTS……46

5.1 SNR COMPARISON……47

5.1.1 SNR Comparison of Solo Piano……...49

5.1.2 SNR Comparison of Drum Solo……50

5.1.3 SNR Comparison of Rock Song……52

5.2 DISTORTION COMPARISON……54

5.2.1 Distortion Comparison of Solo Piano……56

5.2.2 Distortion Comparison of Solo Drums……...50

5.2.3 Distortion Comparison of a Rock Song……..57

5.3 EFFECT OF IMPULSIVE NOISE ON BEAT TRACKING ALGORITHM……58

5.4 FILTERING AUTHENTIC NOISY SOURCE MATERIAL……...59

5.5 SUBJECTIVE LISTENING TEST……62

5.5.1 Interpretation of Results……64

5.5.2 Analysis of Other Results……..65

6 CONCLUSION……..67

7 FUTURE STUDIES……..69

8 BIBLIOGRAPHY……70

9 APPENDIX A: BROADBAND FILTER CODE……...71

9.1 FILTER WITH HARMONIC PREDICTION……...71

9.2 FILTER WITH HARMONIC PREDICTION AND PASSING WINDOWS THAT CONTAIN BEATS…….....75

9.3 FILTER WITH THRESHOLD RE-ESTIMATED ONE WINDOW AFTER BEAT, WITH HARMONIC PREDICTION, AND PASSING WINDOWS THAT CONTAIN BEATS……...82

9.4 PEAK DETECTION CODE……..89

9.5 CODE FOR MASKING THRESHOLD……...89

9.6 CODE FOR BEAT LOCATION……...91

List of Tables

TABLE 5-1: SIGNAL TO NOISE RATIO COMPARISON FOR BOTH THE ORIGINAL FILTER AND THE FILTER WITH HARMONIC PREDICTION……50

TABLE 5-2: SIGNAL TO NOISE RATION COMPARISON OF BOTH THE ORIGINAL FILTER AND THE FILTER THAT PASSES THE BEATS……...52

TABLE 5-3: COMPARISON OF SIGNAL TO NOISE RATIOS FOR DIFFERENT FILTERS USING A ROCK SONG EXCERPT……53

TABLE 5-4: COMPARISON OF SIGNAL TO NOISE RATIOS FOR DIFFERENT FILTERS UPDATED ONE WINDOW AFTER THE BEAT VS. UPDATING EVERY WINDOW……54

TABLE 5-5: COMPARISON OF DISTORTION BETWEEN REGULAR FILTER AND FILTER WITH HARMONIC PREDICTION FOR A SOLO PIANO (PERCUSSION-FREE) EXCERPT……..56

TABLE 5-6: COMPARISON OF DISTORTION BETWEEN REGULAR FILTER AND FILTER THAT PASSES THE SIGNAL ON THE BEAT……..57

TABLE 5-7: COMPARISON OF DISTORTION BETWEEN DIFFERENT FILTERS……58

TABLE 5-8: LISTENING TEST RESULTS FOR EACH LISTENER……...63

TABLE 5-9: AVERAGE TEST RESULTS FOR ALL LISTENERS……..63

List of Figures

FIGURE 2-1: 20 MILLISECOND EXCERPT OF AUDIO SIGNAL ("HAITIAN DIVORCE BY STEELY DAN)……..……3

FIGURE 2-2: POWER SPECTRUM OF THE SAME WAVEFORM IN FIGURE 2-1...4

FIGURE 2-3: FLETCHER-MUNSON EQUAL LOUDNESS CURVES...……..7

FIGURE 2-4: CRITICAL BANDS OF THE EAR: (A) BAND EDGES OF THE BARK SCALE. (B) RELATIONSHIP BETWEEN HERTZ AND BARK FREQUENCIES……...8,9

FIGURE 2-5: SIMULTANEOUS MASKING…….10

FIGURE 3-1: BLOCK DIAGRAM OF SPECTRAL SUBTRACTION SYSTEM [5]...14

FIGURE 3-2: BLOCK DIAGRAM OF WIENER FILTER…….15

FIGURE 3-3: THE ENCODING AND DECODING PROCESS [7]……..19

FIGURE 3-4: FILTER FREQUENCY RESPONSE [7]……..20

FIGURE 3-5: SLIDING OF THE FILTER [7]…….20

FIGURE 4-1: NOISE FILTER BLOCK DIAGRAM……...23

FIGURE 4-2: PLOT OF THE STPS AND THE MEDIAN FILTER……...26

FIGURE 4-3: PLOT OF THE STPS AND THE AVERAGE FILTER……26

FIGURE 4-4: PLOT OF THE STPS AND THE THRESHOLD CURVE……...27

FIGURE 4-5: BLOCK DIAGRAM OF CALCULATION OF MASKING THRESHOLD……31

FIGURE 4-6: MODEL OF SPREADING FUNCTION B(Z) PLOTTED AGAINST NORMALIZED BARK SCALE……...32

FIGURE 4-7: PLOT OF THE STPS AND THE MASKING THRESHOLD……..33

FIGURE 4-8: TOP SHOWING THE ORIGINAL SPECTRUM, THE FILTER SPECTRUM, AND THE FILTERED SPECTRUM WITHOUT HARMONIC PREDICTION. BOTTOM SHOWING THE POWER SPECTRUM AND THE DIFFERENT COMPONENTS OF THE THRESHOLD……..36

FIGURE 4-9: TOP SHOWING THE ORIGINAL SPECTRUM, FILTER RESPONSE, AND FILTERED SPECTRUM WITH HARMONIC PREDICTION. BOTTOM SHOWING THE POWER SPECTRUM AND THE COMPONENTS THAT MAKE UP THE THRESHOLD……37

FIGURE 4-10: THE ENERGY FLUX FOR A ROCK SONG……40

FIGURE 4-11: IMPULSE SIGNAL REPRESENTING TEMPO OF 90 BPM……...41

FIGURE 4-12: PLOT SHOWING THE DIFFERENT PIECES OF THE RECURSIVE SCORING EQUATION DISCUSSED ABOVE……...43

FIGURE 4-13: SHOWING WINDOW RE-ALIGNMENT……45

FIGURE 5-1: FREQUENCY SPECTRUM OF ONE SEGMENT SHOWING THE ORIGINAL SPECTRUM (BLUE) AND THE FILTERED SPECTRUM (GREEN)……60

FIGURE 5-2: SPECTROGRAM OF NOISY SONG……...60

FIGURE 5-3: SPECTROGRAM OF FILTERED SONG……61

1 Introduction

Filtering audio signals corrupted by broadband noise presents a unique problem. The very nature of broadband noise is that its effects are not confined to a narrow, easily extracted bandwidth. Rather, its influence is spread out across many frequencies. Therefore most basic types of filters *i.e.* low pass, high pass, band pass, and notch filters will not yield acceptable noise reduction without causing severe distortion of the frequency content of the underlying audio signal.

A common method for broadband noise reduction is *spectral subtraction* [1]. In spectral subtraction the corrupted signal is modeled as from the result of an additive process. The noise components are added to the original uncorrupted signal to yield the noise corrupted output signal. Therefore the original signal can theoretically be recovered by simply subtracting the noise from the corrupted signal. This subtraction is carried out in the frequency domain, using an estimate of the noise magnitude spectrum. There are however drawbacks to this technique of noise reduction. One major drawback is the creation of perceptually annoying "musical noise" [1] – residual random noise peaks left behind after noise subtraction that are concentrated at particular frequencies, leading to the perception of brief tones that change randomly in frequency. The spectral subtraction technique also relies heavily on acquiring a good estimate of the noise magnitude spectrum which requires gaps of silence in the signal where there is only noise present. In speech enhancement this is generally not a problem as there are pauses and breaks in speech. In musical audio there are rarely any periods in a song that are devoid of vocal or instrumental content.

The noise reduction technique used in this thesis approaches the problem by identifying and preserving the signal content in a song, rather than estimating and eliminating the noise component. The method is based on selecting frequency bins in the magnitude spectrum of the audio signal that clearly contain components of the original signal [1]. These selected bins are left unprocessed, while attenuation is applied on the other noisy spectral bins according to psycho-acoustical rules. This basic technique has already been developed and demonstrated for speech processing in reference [1]. This technique requires multiple passes through each segment being filtered and therefore cannot be applied for real-time signal processing. Improvements on this technique were explored, including using harmonic structure to distinguish musical signal components from the noise, harmonic prediction to maintain the fidelity of musical signals, and the use of tempo to determine filter lengths and appropriate times to re-estimate the harmonic structure of the musical signal. The main application for this filtering technique is in the restoration of audio recordings corrupted by broadband noise.

2 Fundamentals

2.1 Audio Signals

Audio signals are complex waveforms composed of combinations of different musical notes (different fundamental frequencies) with unique harmonic content determined by the particular instruments or voices creating the notes.

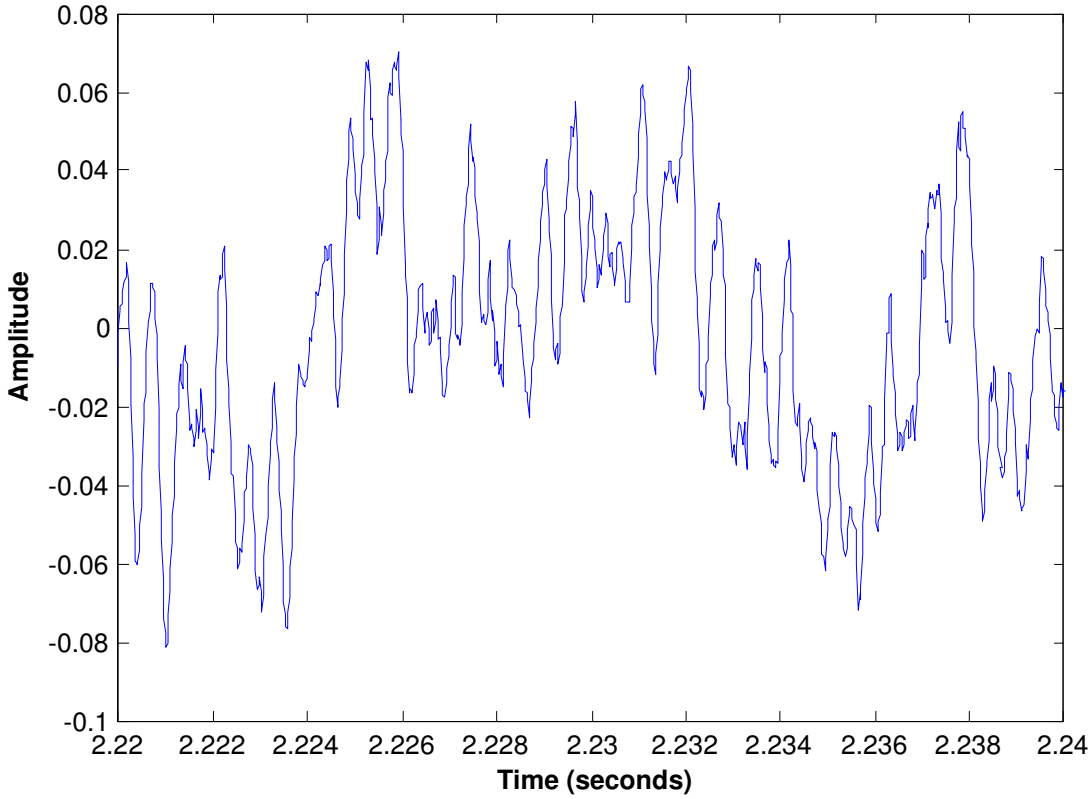


Figure 2-1: 20-millisecond excerpt of an audio signal ("Haitian Divorce" by Steely Dan)

The harmonic content of a signal is nearly impossible to discern from the time domain waveform. Taking the Fourier transform of the signal and looking at the power spectral density reveals the frequency components that make up the signal.

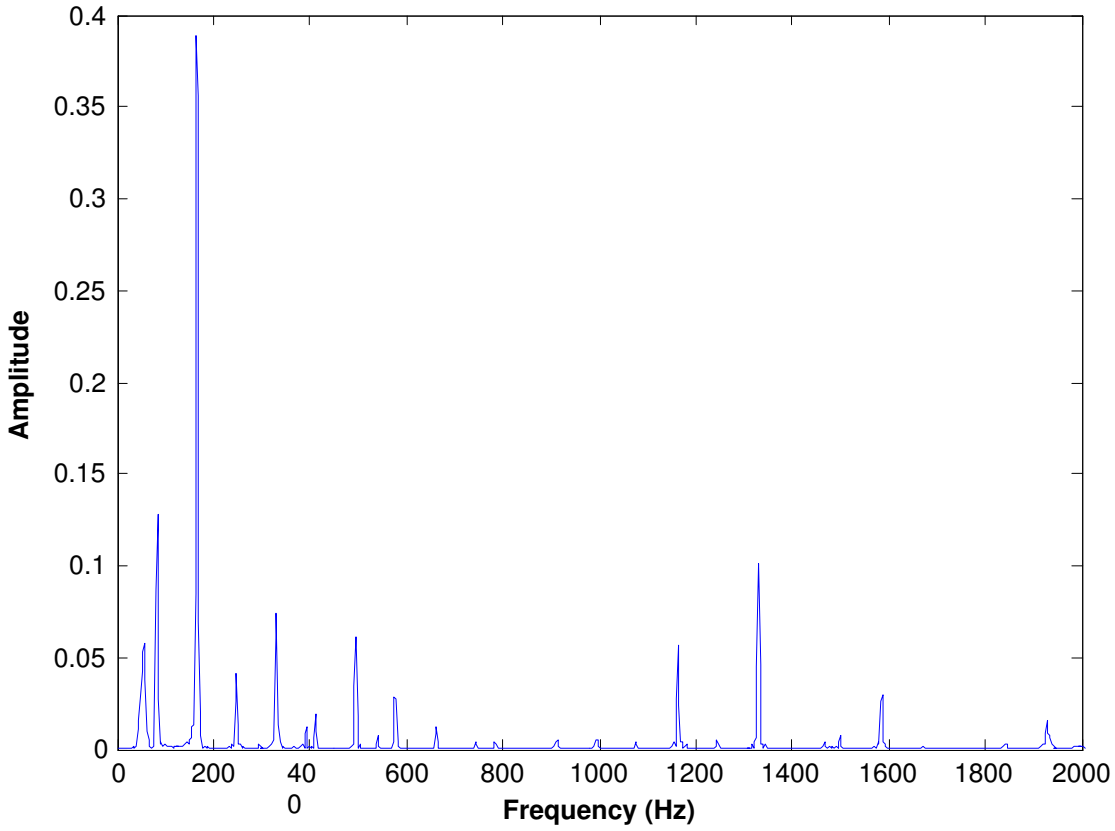


Figure 2-2: Power Spectrum of the same waveform in Figure 2-1

Figure 2-2 shows that the time domain waveform of Figure 2-1 is composed of components from many different frequencies. The largest peak at 165 Hz has a second harmonic at 330 Hz, a third at 495 Hz, and a fourth at 660 Hz all of which can be seen in Figure 2-2. This very basic technique of looking at how the power of the signal is distributed in equally spaced harmonics across the frequency spectrum is the basis for distinguishing signal content from noise.

2.2 Decision Threshold

An adaptive threshold is used to decide which frequency bins clearly contain components of the original signal and which bins contain noise. The threshold is calculated every frame and therefore changes from frame to frame. The decision rule is as follows: if a frequency bin contains enough power that it is above the threshold it is considered a component of the original signal and it is retained. If the signal level in a frequency bin is below the threshold, it is suppressed using a set of psycho-acoustic rules that will be elaborated later. Ideally the threshold level would sit slightly above the noise floor, therefore suppressing the noise and passing the larger frequency peaks that are associated with the harmonic content of the original signal. However if a harmonic component does not posses enough power to be above the threshold it will be suppressed, which would lead to a less faithful reproduction of the original signal.

2.2.1 Median and Moving Average Filtering

The threshold that will be used to separate harmonic content from noise is computed using both median and moving average filters. A median filter goes sample by sample through the signal and replaces the present sample with the median of its neighboring samples. The number of samples used to determine the median is referred to as the window. The window is shifted sample by sample through the entire signal. Given a discrete set of samples the median filter is defined below.

$$y(n)=median(\ x[n-k],\ldots,\ x[n],\ldots,\ x[n+k]\ ) \quad (2.1)$$

where $N = 2k + 1$ is the window length.

The moving average filter uses a sliding window similar to the median filter and calculates the average in that window; replacing the present sample in the center of the window with the average of the neighboring samples. The equation for a moving average is defined below.

$$y[i] = \frac{1}{M} \sum_{j=0}^{M-1} x[i+j] \quad (2.2)$$

where x[ ] is the input signal, y[ ] is the output signal, and M is the number of points in the window.

2.3 Psychoacoustics

How a person perceives sound should be taken into account in the technique used to suppress the noise when filtering a signal. Excessive filtering can introduce undesirable sound artifacts into the signal. Better results can be achieved if the noise is suppressed just enough so that it was no longer audible. Ultimately, all that matters is how the listener perceives the filtered signal. The psychoacoustic phenomena that are important to the suppression used in this thesis are the *absolute threshold of hearing, critical band frequency analysis,* and *simultaneous masking.* The ultimate goal is to determine a masking threshold for each segment of the input audio signal that noise components need to be attenuated below to become inaudible.

2.3.1 Absolute Threshold of Hearing

The absolute threshold of hearing is characterized by the amount of energy needed in a pure tone so that it can be detected by a listener in a noiseless environment [2].

This threshold is frequency dependent. The ear does not have a flat frequency response; it is most sensitive in the 1 to 5 kHz range. Through subjective testing curves have been derived that show the ear's sensitivity as a function of frequency. Fletcher and Munson curves are the most common plots for showing the frequency dependence of ear sensitivity. Figure 2-3 shows Fletcher-Munson equal loudness curves. Each curve on the plot shows a range of frequencies and sound intensity levels that are perceived to be equally loud. The dashed curve represents the absolute threshold of hearing. When processing a signal any components determined to be noise that lie below the dashed curve can immediately be discarded since the listener cannot hear them.

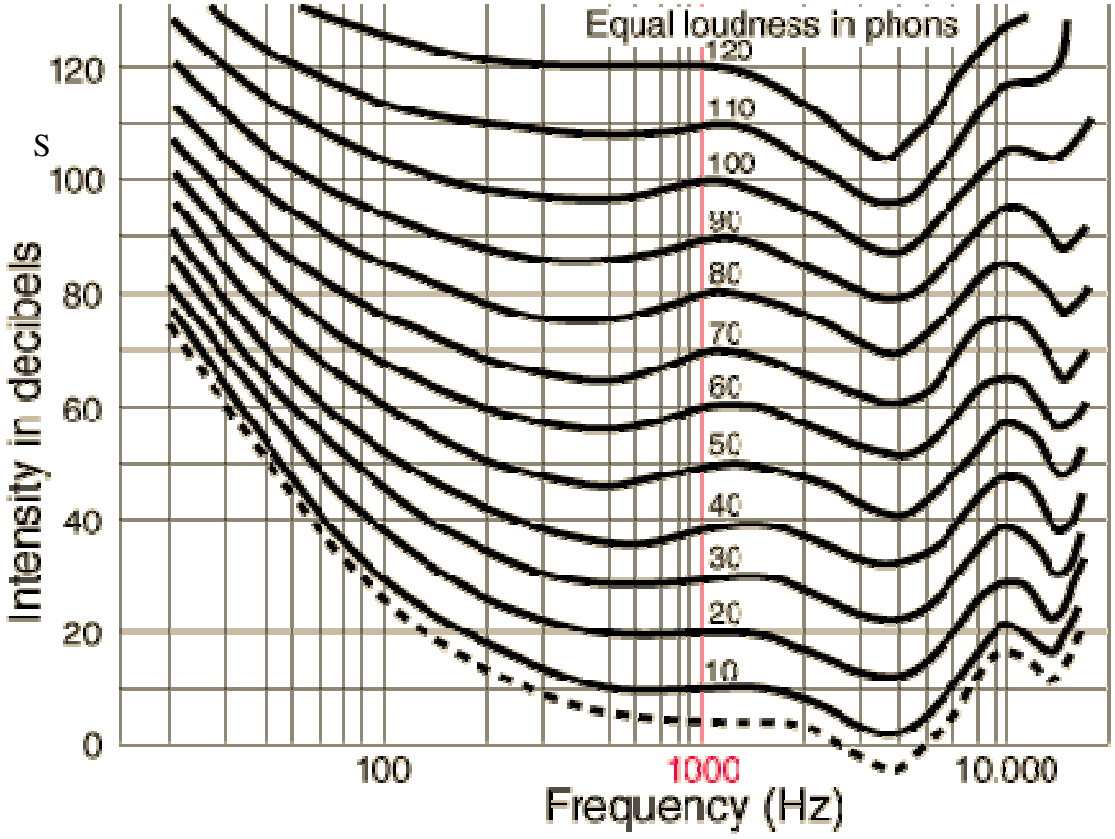
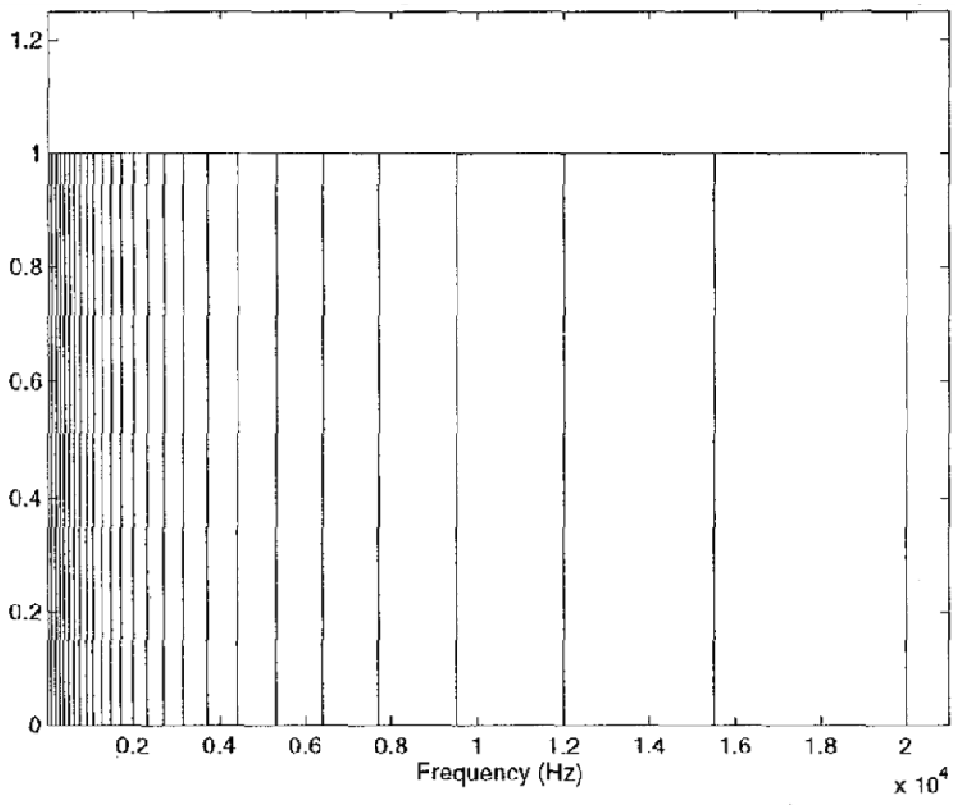


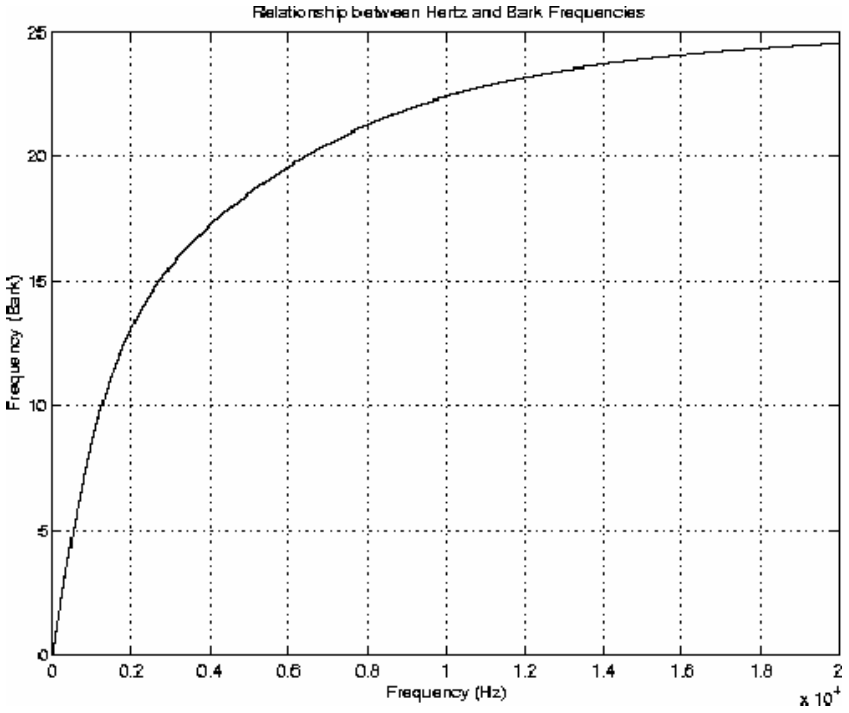
Figure 2-3: Fletcher-Munson equal loudness curves

2.3.2 Critical Band Frequency Analysis

The way a person's ear interprets the spectral content of a signal is the basis of critical band frequency analysis. The basilar membrane in the ear can be thought of as a filter bank of overlapping band pass filters. The bandwidths of these band pass filters are not all equal. They vary from a few hundred hertz at low frequencies to thousands of hertz at higher frequencies. The psycho-acoustical scale that is used in this processing algorithm is the Bark scale. The Bark scale was proposed by Eberhard Zwicker in 1961 and is named after Heinrich Barkhausen who proposed the first subjective measurements of loudness [3]. Through his studies, Zwicker developed a scale that has 25 critical bands. Figure 2-4(a) shows how the edges of each critical band correspond to frequency.



(a)



(b)

Figure 2-4: Critical Bands of the ear: (a) Band edges of the Bark scale. (b) Relationship between temporal (Hertz) and Bark frequencies

If two tones within the same critical band were played at the same intensity level the ear would be unable to resolve the two separate tones. The critical bandwidths at low frequencies are narrow. Therefore, two low frequency tones would only have to be separated by a few hundred hertz for the ear to resolve two separate tones. At higher frequencies, the tones would have to be thousands of hertz apart to be resolved.

2.3.3 Simultaneous Masking

Masking refers to the process where one sound is rendered inaudible because of the presence of another sound [2]. Masking is another aspect of critical band frequency analysis. For example, the critical bandwidth for a 5 kHz sine wave is 900 Hz wide. If there was a 900 Hz wide noise source centered at 5 kHz present at the same time as

a 5 kHz sine wave signal, the noise would only be audible if it was greater than or equal in amplitude to the masking threshold established by the 5kHz sine wave signal.

The term *simultaneous masking* simply refers to the fact that two sounds are played at the same time and one sound is hidden by the other. In simultaneous masking there are two cases to consider. Within critical bands there are the phenomena of *tone masking noise* and the reverse, *noise masking tone*. Tone masking noise is the only case of concern for the noise suppression used in this thesis. In the case of tone masking noise, a tone at the center of a critical band will mask any noise within that same critical band, if the noise amplitude is less than a defined threshold level that depends on the frequency and strength of the tone.

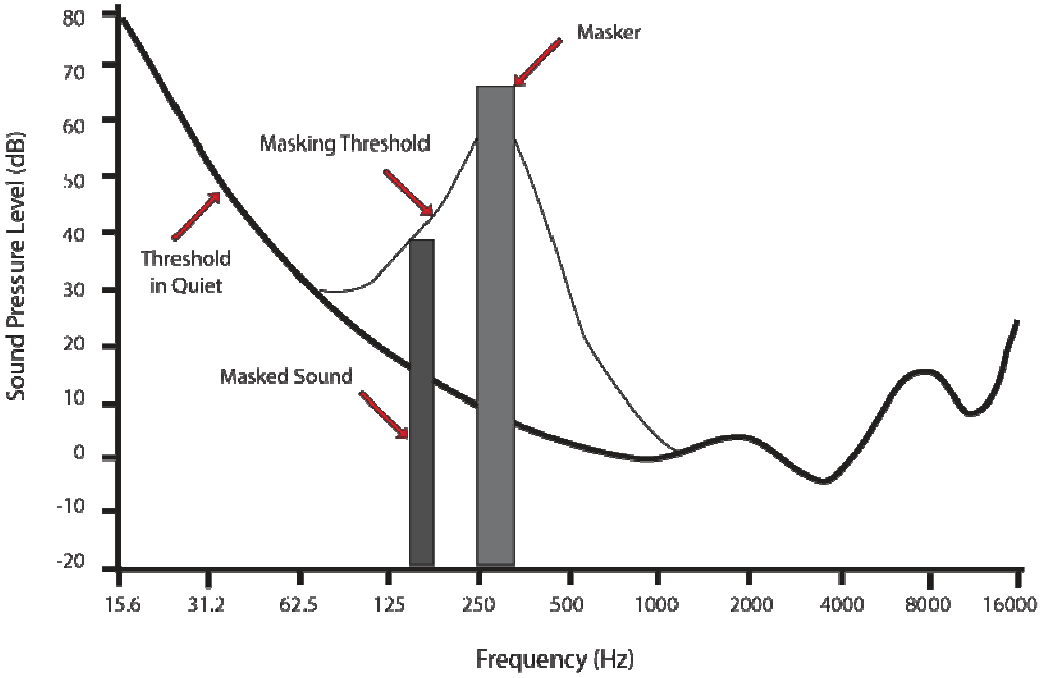


Figure 2-5: Simultaneous masking

Figure 2-5 illustrates what is happening when simultaneous masking is taking place. Figure 2-5 shows the masked sound pressure level below the shifted masking threshold; therefore it will not be audible to the listener.

Another phenomenon that will be accounted for in the suppression algorithm is *spread masking*. When simultaneous masking occurs, a masker centered within one critical band has some predictable effect on detection thresholds in other critical bands [2]. The analytic expression used to describe spread masking is given as:

$$B_{dB}(z) = 15.91 + 7.5(z + 0.474) - 17.5\sqrt{1 + (z + 0.474)^2} \quad (2.3)$$

where z has units of Barks and B_{dB}(z) expressed in dB is an approximation of the basilar membrane spreading function.

3 Existing Noise Suppression Algorithms

3.1 Spectral Subtraction

3.1.1 Introduction

Spectral Subtraction is one of the older techniques used to suppress broadband noise. It was originally developed for use with speech signals. There are certain aspects of spectral subtraction that carry over to the algorithm used in this thesis, such as how the phase of the signal is dealt with.

3.1.2 Spectral Subtraction Theory

In spectral subtraction the corrupted signal is modeled as the result of an additive process. The noise is simply added to the original signal, yielding the corrupted signal. Therefore the noise can simply be subtracted from the corrupted signal, leaving the original signal intact. This subtraction is performed in the frequency domain rather than the time domain. The additive process is modeled by the following equation.

$$x(k) = s(k) + n(k)$$

Where $n(k)$ is the windowed noise signal, $s(k)$ is the windowed original signal, and $x(k)$ is their sum. Taking the Discrete-Time Fourier Transform of the above equation gives

$$X(e^{jw}) = S(e^{jw}) + N(e^{jw})$$

$$\left|X(e^{jw})\right| e^{j\theta_x(e^{jw})} = S(e^{jw}) + \left|N(e^{jw})\right| e^{j\theta_N(e^{jw})}$$

The spectral subtraction filter $H(e^{jw})$ is calculated to suppress the noise spectrum component $N(e^{jw})$, which can be readily estimated from signal and noise measurements [5]. The magnitude of the noise spectrum at each frequency $|N(e^{jw})|$ is estimated by an

average of measured spectral magnitude values $\mu(e^{jw})$ computed by Fourier Transforming time-domain sample windows measured during non-speech segments, when only noise is present. The phase of the measured noise spectrum $\theta_N(e^{jw})$ is assumed to be the same as the phase of the corrupted signal spectrum $\theta_X(e^{jw})$, rather than the random phases from non-speech time samples. The resulting uncorrupted signal estimate is given by

$$\hat{S}(e^{jw}) = \left[\left|X(e^{jw})\right| - \mu(e^{jw})\right] e^{j\theta_X(e^{jw})}$$

$$\mu(e^{jw}) = E\{\left|N(e^{jw})\right|\}$$

As in the above signal estimate, retaining the original phase of the corrupted signal and recombining it with the noise-filtered signal magnitude is the approach used in the filtering scheme of this thesis. The obvious problem with this technique is a heavy reliance on accurate estimation of the noise spectrum magnitude. The background noise environment is assumed to remain locally stationary to the degree that its spectral magnitude expected value just prior to speech activity equals its expected value during speech activity [5]. If the noise environment changes to a new stationary state there must be enough time to estimate a new background noise spectral magnitude expected value before speech activity recommences [5]. This is one of the main reasons this technique is unattractive for use with audio signals. Most audio contains continuous musical signals with no periods where only noise is present. With no ability to re-estimate the noise spectrum average, the filter results deteriorate as the noise spectrum changes over time.

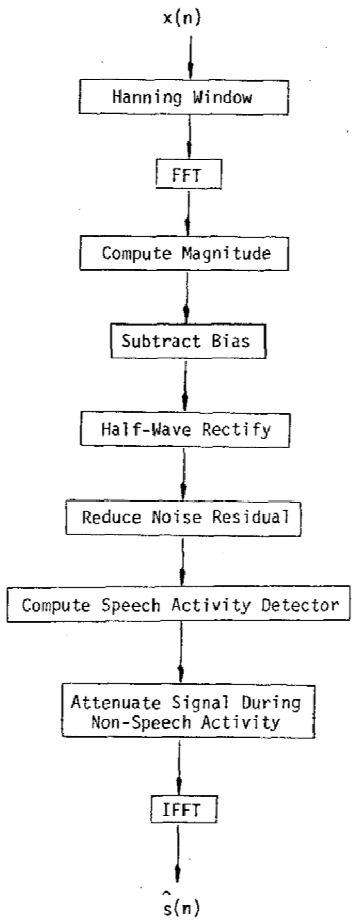


Figure 3-1: Block diagram of spectral subtraction system [5]

3.2 Wiener Filtering

3.2.1 Introduction

The Wiener filter is filtering technique proposed by Norbert Wiener and published in 1949. The idea behind the Wiener filter is to reduce the amount of noise present in a signal by comparing it with the desired noiseless signal. The Wiener filter is a statistically designed filter that assumes the inputs are stationary. Since the inputs are assumed to be stationary the Wiener filter is not an adaptive filter.

3.2.2 Wiener Filter Theory

Similar to the spectral subtraction technique the corrupted signal is modeled as an additive process

$$x(n) = d(n) + n(n)$$

where the original signal $d(n)$ plus the noise $n(n)$ equals the corrupted signal $x(n)$. The criterion selected for optimizing the filter coefficients is the minimization of the mean square error.

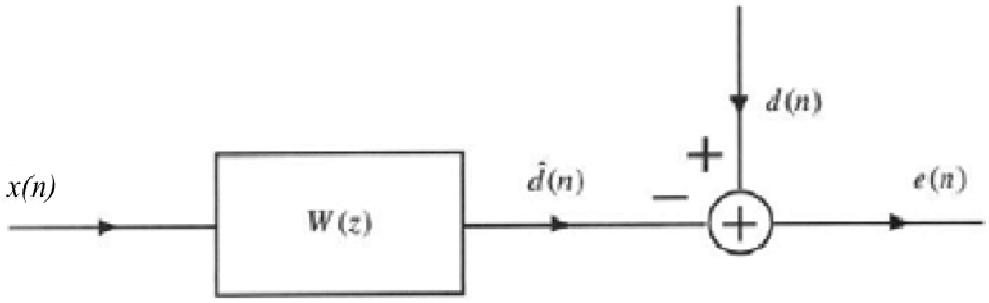


Figure 3-2: Block diagram of Wiener Filter

The signal $\hat{d}(n)$ in Figure 3-2 is the estimate of the true signal *d(n)*. The signal $e(n)$ is the error signal which is the modeled as the difference between the filter output and the desired signal.

$$e(n) = d(n) - \hat{d}(n)$$

The output of the filter is

$$\hat{d}(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

where $h(k)$ is the impulse response of the Wiener Filter. In matrix form with

$$x(n) = \begin{bmatrix} x(n) \\ x(n-1) \\ . \\ . \\ x(n-M) \end{bmatrix} \quad \text{and} \quad h(n) = \begin{bmatrix} h_0(n) \\ h_1(n) \\ . \\ . \\ h_M(n) \end{bmatrix}$$

Now the error squared can be written as

$$e^2(n) = \left[d(n) - h^T(n)x(n)\right]^2$$

expanding this equation leads to

$$e^2(n) = d^2(n) + \left[h^T(n)x(n)\right]^2 - 2d(n)h^T(n)x(n)$$

Now taking the expected value, we have a mean squared error of

$$E\left[e^2(n)\right] = E\left[d^2(n)\right] + h^T(n)E\left[x(n)x^T(n)\right]h(n) - 2h^T(n)E\left[d(n)x(n)\right]$$

$R_{xx}(n) = E\left[x(n)x^T(n)\right]$ is the autocorrelation matrix of the input sequence x(n).

$P = E\left[d(n)x(n)\right]$ is the cross correlation matrix of the desired and the input sequences.

Therefore substituting R and P in the mean squared error equation

$$E\left[e^2(n)\right] = P_d(n) + h^T(n)Rh(n) - 2h^T(n)P$$

Now in order to minimize the mean squared error we take its derivative and set it equal to zero.

$$\frac{\partial\left(E\left[e^2(n)\right]\right)}{\partial h} = 0$$

This leads to $0 + 2h^T R - 2P^T = 0$

And solving for *h* gives the Wiener solution

$$h = R^{-1}P$$

The main problem with this technique is its dependence on knowing the desired signal in order to form the cross correlation matrix P. Obviously in a real life situation the

corrupted signal will be the only signal present. Therefore an estimate of the desired signal needs to be obtained from the corrupted signal. Another problem is the audio signal and possibly the noise are neither stationary nor ergodic processes, which means the probability distributions can not be obtained by examining the long term time behavior of each signal [5]. According to [5] there is a way around these problems which leads to the Wiener transfer function as a ratio of the a priori SNR to one plus the a priori SNR. A derivation from [6] follows with the observed corrupted speech $y(n)$ equal to the clean speech $s(n)$ plus the noise $b(n)$. The processing is done on a frame-by-frame basis in the spectral domain [6]. The short time Fourier transform magnitude of the speech is

$|S_d(m,f)|$ where m is the frame index and f is the frequency.

$|S_d(m,f)|^2 = H_{opt}(m,f)|Y(m,f)|^2$ where H is the denoising filter given by

$$H_{opt}(m,f) = \frac{SNR_{apriori}(m,f)}{1+SNR_{apriori}(m,f)}$$ where $SNR_{apriori}$ is the a priori signal to noise ratio

$$SNR_{apriori}(m,f) = (1-\tau)P(SNR_{post}(m,f)) + \tau\frac{|H_{opt}(m-1,f)\|Y(m-1,f)|^2}{\Gamma_n(m,f)}$$

where τ is a real constant, P(x)=1/2(x+|x|), $\Gamma_n(m,f)$ is the noise power spectrum estimate and SNR_{post} is the a posteriori signal to noise ratio.

$$SNR_{post}(m,t) = \frac{|Y(m,f)|^2}{\Gamma_n(m,f)} - 1$$

The time domain denoised speech can then be obtained by taking the inverse transform.

$$s_d(k) = IFFT\left[|S_d(m,f)|e^{j\arg(Y(m,f))}\right]$$

3.3 Dolby Noise Reduction

3.3.1 Introduction

Dolby noise reduction is a series of noise reduction systems used for analog tape recordings developed by Dolby Labs in the 1960's. They were developed to combat the broadband noise known as tape hiss inherent to recording on magnetic tape. Tape hiss is broadband noise that is present due to the size of the magnetic particles used to make the tape. There are ways to reduce the hiss by use of finer magnetic particles or increasing the amount of tape used per second. However these changes are not compatible with the established standard for studios or home tape machines.

3.3.2 Dolby B

Dolby B was the first noise reduction system the company introduced for use with consumer tape machines. The primary method for improving signal to noise ratio is the same for all the different Dolby systems, but the details of how the signal is broken up and filtered is what differs from system to system. The overall idea is that during recording the quieter high frequency passages are boosted to get their level above that of the tape hiss. Loud passages that already hide the tape hiss are not altered. This process is referred to as *encoding* [7]. During playback the exact opposite process referred to as *decoding* takes place. The soft high frequency sounds are lowered back to their original level. The quieter signals get boosted before they reach the tape and are mixed with the tape hiss. This means that during playback when the boosted signal is brought back down to its original level, the tape hiss is also reduced by the same amount [7]. This is illustrated in figure 3-3.

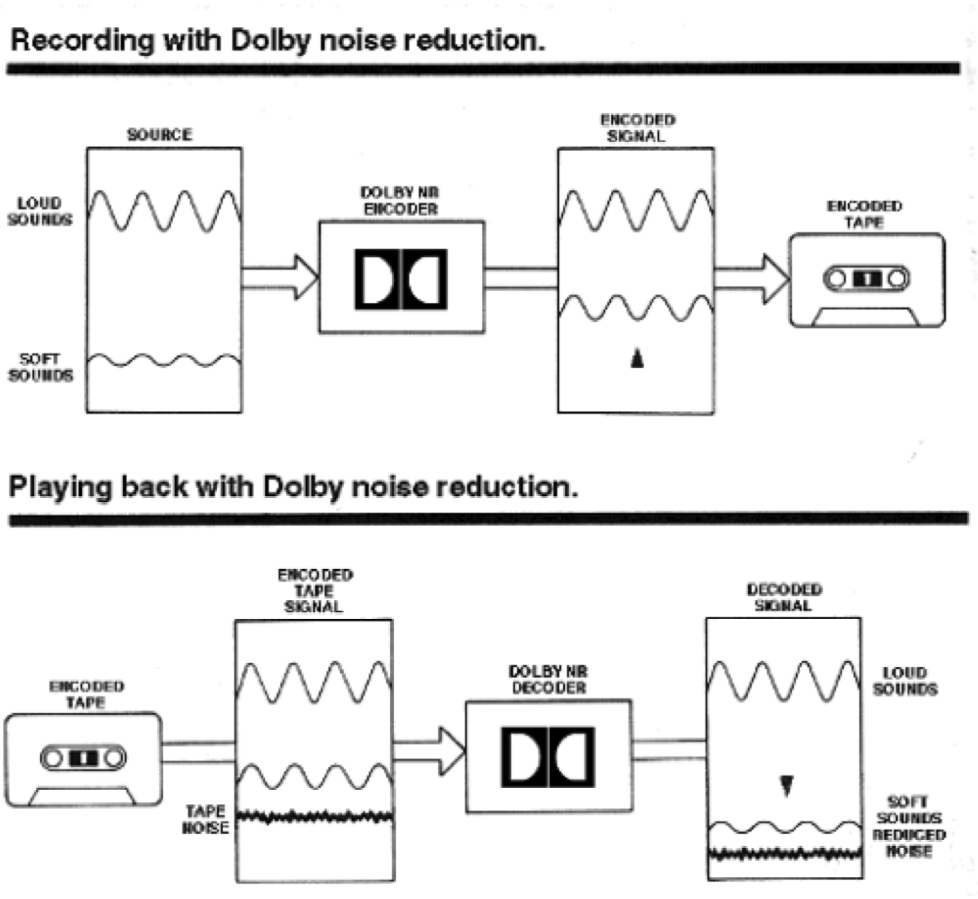


Figure 3-3: The encoding and decoding process [7]

3.3.2.1 Filtering

As was mentioned earlier, what distinguish the different Dolby noise reduction systems are the specifics of filtering the audio signal. In Dolby B there is a single sliding compression-expansion band of frequencies [7]. This filter is essentially a high pass filter that can slide along the frequency axis boosting different frequencies and leaving others unchanged. In the case of Dolby B, the filter can slide from about 300 Hz all the way out to 20,000 Hz and provide a maximum gain of 10dB. Figure 3-4 illustrates how only softer high frequency signals are boosted and louder signals are unaltered.

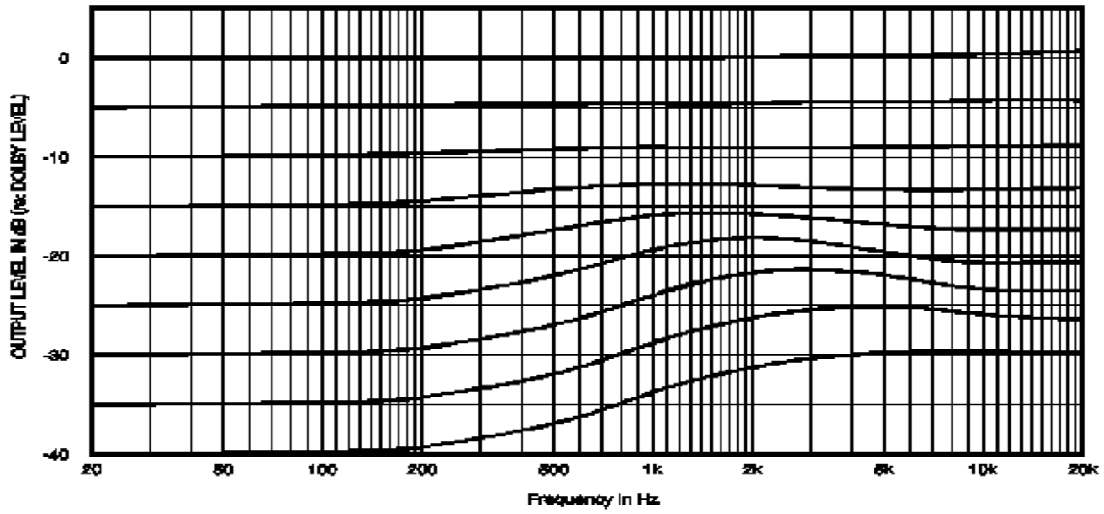


Figure 3-4: Filter frequency response [7]

If a signal level is very low or contains no treble frequencies the filter slides to the lowest frequency point, which gives a maximum noise reduction of 10 dB above 4000 Hz. As the filter slides upward, less and less of the spectrum is affected. The filter will only slide up once the sounds being recorded are loud enough to hide the noise on their own. The filter is then adjusted fast enough to follow the content of the music. Figure 3-5 illustrates how a loud dominant frequency is unaltered by the filter while it continues to reduce noise at higher frequencies.

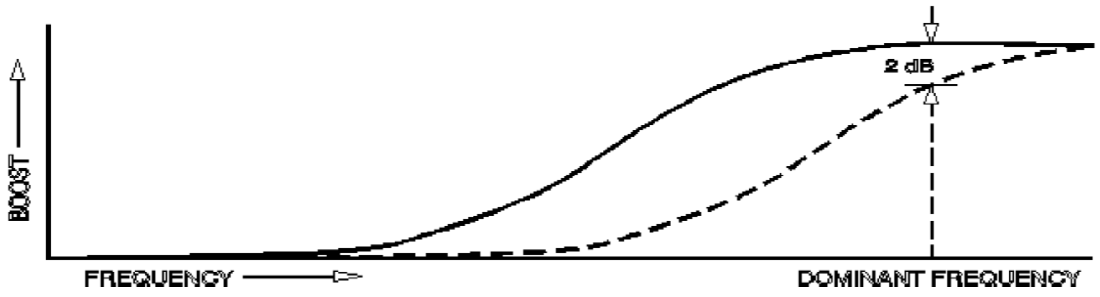


Figure 3-5: Sliding of the filter [7]

In order to achieve perfect restoration of the original signal the decoder must track as closely as possible the locations of the sliding bands used in recording. Mistracking can

occur, but its effects may not be audible. Due to the limitations in gain, Dolby B is not very susceptible to audible mistracking [7].

4 Broadband Noise Reduction Algorithm

The obvious challenge of eliminating broadband noise is its presence at all frequencies, including those containing components of the audio signal that should be preserved. Therefore simple linear filters will not suffice for reducing the noise. The algorithm used in this project looks at the problem as finding a decision boundary to classify each spectral frequency bin as either desired signal to retain or noise to eliminate. Improvements to this basic algorithm will then be made by restoring more harmonic content to the filtered signal and using tempo analysis to increase filtering efficiency and to determine the appropriate processing window length.

The noise model that this algorithm is designed for is broadband noise (white noise), which is common in tape recordings and AM/FM transmission. Similarly, an LP recording may also exhibit broadband noise on playback. However, LP recordings are also subject to impulsive noise (clicks and pops) due to scratches, and mechanical noise imparted onto the signal due to vibrations from the moving parts of a turntable. A broadband noise model would not account for the other types of noise that may be encountered when restoring an LP recording. Ultimately the appropriate noise model depends on the source medium of the audio signal.

The original algorithm presented in [1] defines an adaptive threshold to determine which frequency components are desired signal and which are noise. The desired signal components are left unprocessed and the noise is suppressed. The suppression uses a psychoacoustical modal to calculate a masking threshold. The noise components are then iteratively suppressed until they are below the masking threshold. One of the issues with this approach is that upper harmonics are often eliminated because they do not possess

enough power to be classified as desired signal. Removal of these higher harmonics leads to a muffled, low-pass filtered sound. Adding these harmonics back into the filtered result leads to a crisper more accurate representation of the original uncorrupted signal.

Determining the tempo and location of the beats in a signal is also used to improve efficiency in the filtering process. The harmonic structure of a song is more likely to change on the beats or during sub divisions of the beat period (*i.e.* quarter notes, eighth notes, *etc*.). Therefore it is only necessary to re-estimate and update the noise reduction filter during these times rather than for every signal frame. This reduces the computation time required to filter the signal. Knowing where rhythmic beats occur will also help to determine how to filter percussion, since spectrally percussion often looks like noise and occurs on beats.

4.1 The Broadband Noise Filter

The noise reduction filter can be broken up into two components. They are the adaptive threshold estimate and the suppression function. A diagram of the filter is shown below.

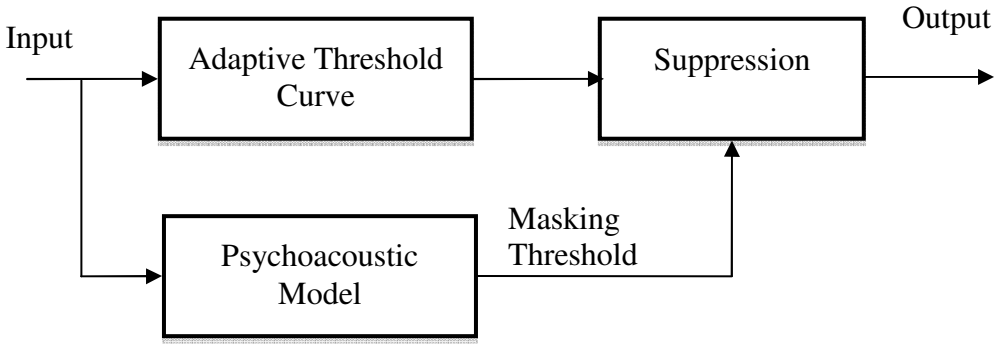


Figure 4-1: Noise Filter Block Diagram

4.1.1 Adaptive Threshold Curve

The signal is first split into non-overlapping segments of length N=8192 samples as specified by [1]. It is important that the window length be short enough to capture and isolate one musical event. This way an accurate decision threshold can be calculated for the content of the present musical segment being processed. The power spectrum for each segment is then calculated by taking the magnitude squared of the Discrete Fourier Transform (DFT) giving the Short Time Power Spectrum (STPS).

STPS:

$$Y[k] = \left| \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \right|^2, \quad k = 0,1,2,....,N-1$$

where x[n] is the sampled input signal.

The adaptive threshold curve is computed from

$$T[f] = \alpha \left[median\left\{ A_p[f] \right\} \right] - \frac{\beta}{2j+1} \sum_{k=f-j}^{f+j} Y_p[k] + \lambda \qquad [1]$$

where $Y_p[f]$ is the STPS of a segment of the audio signal. $A_p[f]$ is a subset of the STPS components with length 2i+1 (where i=100 as specified in [1]) given by

$$A_p[f] = \left\{ Y_p[f-i],...,Y_p[f],...,Y_p[f+i] \right\} \qquad [1]$$

that is median filtered. The second factor in the T[f] threshold curve is simply a scaled average of the STPS over a shorter length 2j+1 (j<i) with j=10. Constants α and β are long term median and short term average scaling factors, respectively [1]. The most important relationship between the median and moving average filters is that the median filter's length (the number of frequency samples considered) must be considerably longer than the moving average filter length. If the median filter is too short, its output will

follow the spectral peaks rather than modeling the background noise floor. If the moving average filter uses too many spectral samples, it will not faithfully pick out the harmonic peaks in the spectrum. λ is a threshold offset value [1]. Lambda should ideally be chosen so that the resulting threshold is positioned just above the noise floor in the spectrum for frequency bins that contain only noise.

The thresholding decision rule for each frequency bin *f* in the audio segment signal spectrum is that *f* contains desired harmonic signal components if the value of the STPS $Y_p[f]$ at that frequency *f* is above the adaptive threshold $T[f]$. Frequency bins with STPS values below the threshold are considered to be noise floor components. The noise reduction filter then operates by retaining the spectral components that exceed the threshold; and suppressing those that fall below the threshold so that

$$\left|\hat{S}[f]\right| = \begin{cases} \left|Y[f]\right|, & \left|Y[f]\right|^2 \geq T[f] \\ H[f]\left|Y[f]\right|, & \left|Y[f]\right|^2 < T[f] \end{cases} \qquad [1]$$

where $H[f]$ is chosen to meet psychoacoustical conditions [1].

The long-term median filter is used to follow the overall shape of the noisy signal power spectrum noise floor. The short-term average is used to significantly lower the threshold level of the frequency bins that contain mostly desired harmonic components. This threshold lowering occurs when the short-term average is subtracted from the long-term median filter. As shown in the following figures, the short-term average value follows the harmonic peaks from musical sources in the power spectrum. The scaled averaging creates a widened window around each harmonic peak. When subtracted from the scaled, long-term median filter value, the resulting threshold level is very low in the vicinity of musical harmonic peaks.

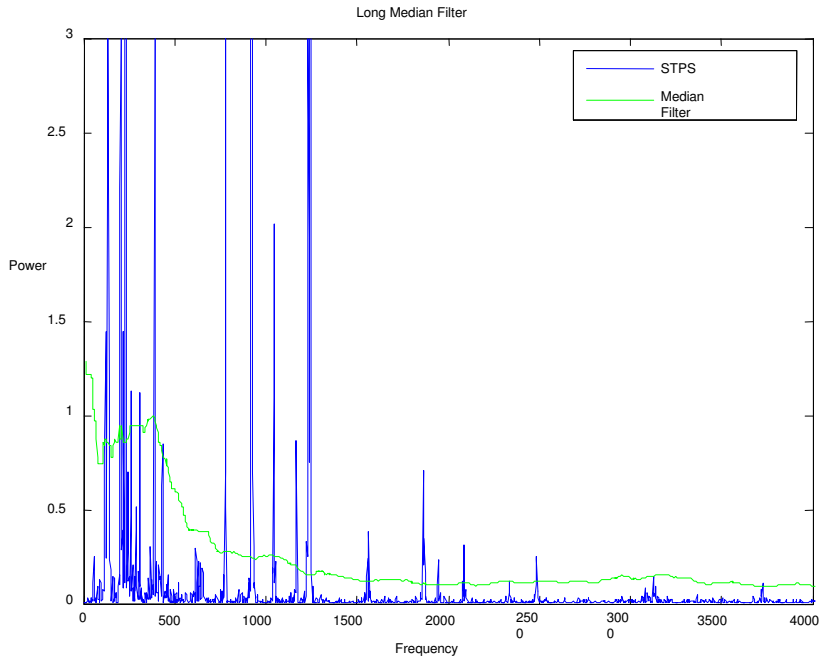


Figure 4-2: Plot of the STPS and the Median Filter

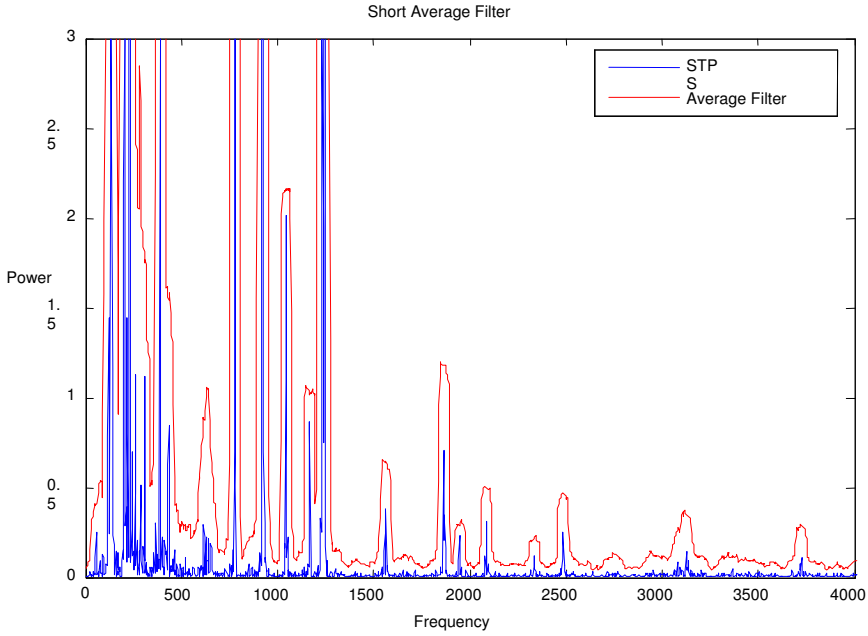


Figure 4-3: Plot of the STPS and the Average Filter

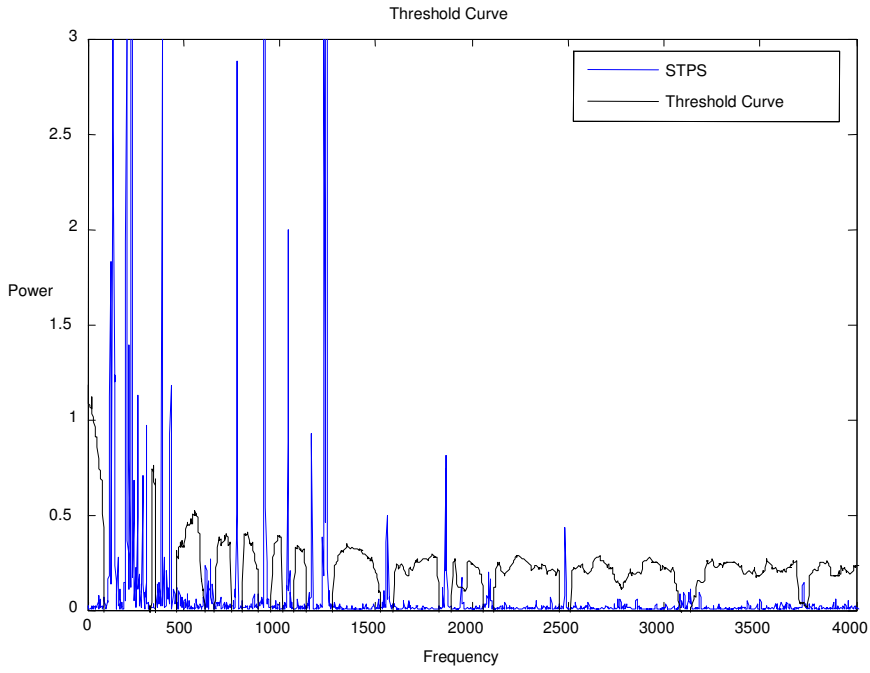


Figure 4-4: Plot of the STPS and the Threshold Curve

4.1.2 Masking Threshold Calculation

The masking threshold is used to determine how severely the noise components need to be attenuated to become inaudible. Noise that is below the masking threshold is not detectable by the ear. The masking threshold is calculated using the Bark scale – a subdivision of the audible range into 24 critical bands. The band value is also known as the critical band rate (CBR) [1]. The steps to calculating the masking threshold are as follows:

Step 1: Conversion of power spectrum frequencies into Bark scale values.

The first step to calculating the masking threshold is to convert the frequencies of each window that are in cycles/second (Hz) into Bark scale frequency, denoted as z. This is done using an approximation for the frequency transformation given by

$$z = 13\tan^{-1}\left(\frac{0.76f}{1000}\right) + 3.5\tan^{-1}\left\{\left(\frac{f}{7500}\right)^2\right\} \qquad [1]$$

where z is the mapped frequency in Barks.

Step 2: Determining the amount of energy in each critical band.

The amount of energy in each critical band forms the basis for calculating the masking threshold. The critical band energies are calculated by simply summing up the square of the short-time spectrum magnitudes over the range of frequencies contained in each critical band.

$$E[z] = \sum_{f=lz}^{hz} Y_p[f] \qquad [1]$$

The summation limts *lz* and *hz* are the lower and upper frequency boundaries assigned to each critical band z.

Step 3: Determine the spread masking level C(z) across critical bands.

This is done by convolving the energy per critical band E(z) with an approximation of the human ear's basilar membrane spreading function. B(z)

$$C(z) = E(z) * B(z) \qquad [1]$$

In dB, the basilar membrane spreading function is given by

$$B_{dB}(z) = 15.91 + 7.5(z + 0.474) - 17.5\sqrt{1 + (z + 0.474)^2} \qquad [1]$$

Step 4: Compute the spectral flatness measure (SFM) and index of tonality.

When determining the masking threshold, two different cases must be considered: noise masking tone (NMT) and tone masking noise (TMN) [1]. Different masking threshold offset values are used for these two cases: 5.5dB below C(z) for NMT, and (14.5+z) dB below C(z) for TMN [1]. The spectral flatness measure (SFM) is used to determine which case is present for each critical band. The SFM is defined as the ratio of the geometric mean (Gm) of the power spectrum to the arithmetic mean (Am) of the power spectrum [1]. The SFM is given in decibels as

$$SFM = 10\log_{10}\left(\frac{Gm}{Am}\right) \quad [1]$$

This is then used to generate the index of tonality given by

$$\delta = \min\left(\frac{SFM}{SFM_{\max}}, 1\right) \quad [1]$$

SFM_{max} is defined as -60dB.

The tonality index will tell us if the frame being analyzed is more tone like or more noise like. The tonality index is used to weight the offset for the masking threshold properly, either towards TMN or NMT. As δ approaches a value of 1, the tone masking noise case is considered to dominate in the particular critical band. Small fractional values of δ conversely, indicate the case of noise masking tone in the band.

Step 5: Calculating the masking threshold offset and the final masking threshold.

The offset for the masking threshold is defined as follows

$$O_{dB}(z) = \delta(14.5 + z) + (1 - \delta)5.5 \qquad [1]$$

This function weights the masking offset values for the two masking cases by the degree to which each case applies in the current critical band. The final masking threshold is then calculated by subtracting the offset from the spread masking threshold C(z) and normalizing the result by the number of samples in each critical band. The equation for the final masking threshold is given as

$$M(z) = \frac{10^{\left(\log_{10} C(z) - \left(\frac{O_{db}(z)}{10}\right)\right)}}{N} \qquad [1]$$

where N is the number of sample points in each critical band.

The masking function in Figure 4-5 is calculated for the same signal frame used in figures 4-2,3,4 with the only difference being that the frequency axis in Figure 4-5 is displayed on a logarithmic scale.

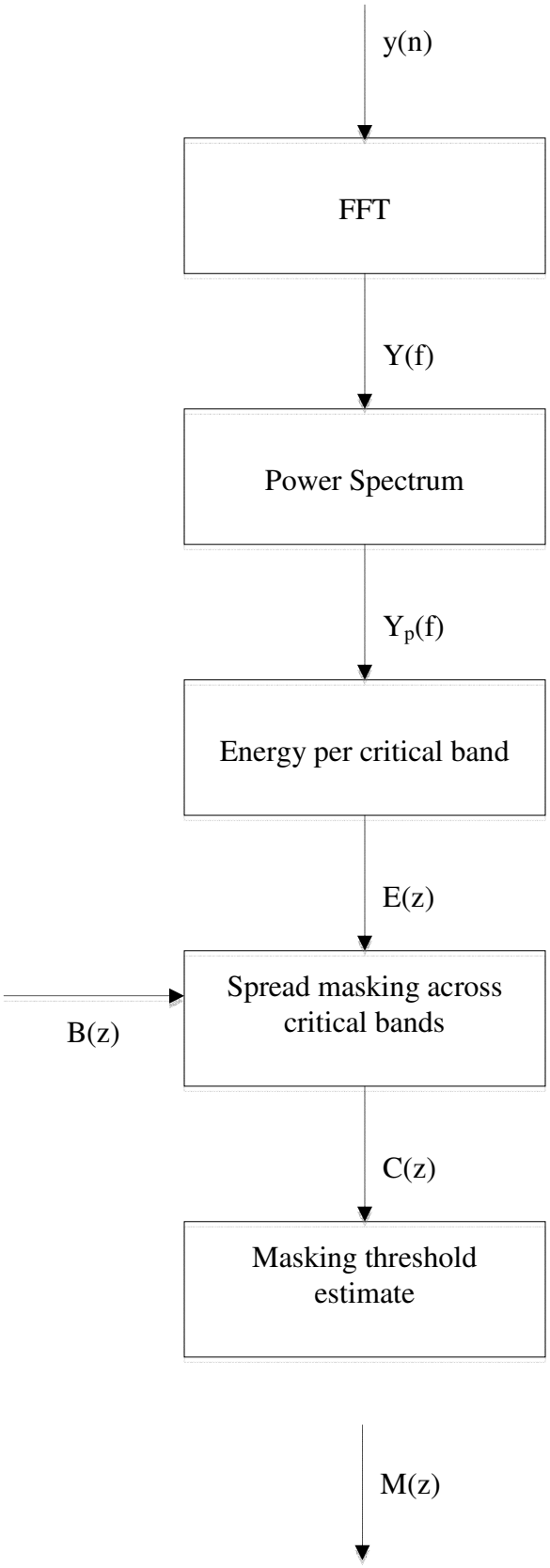


Figure 4-5: Block diagram of calculation of masking threshold

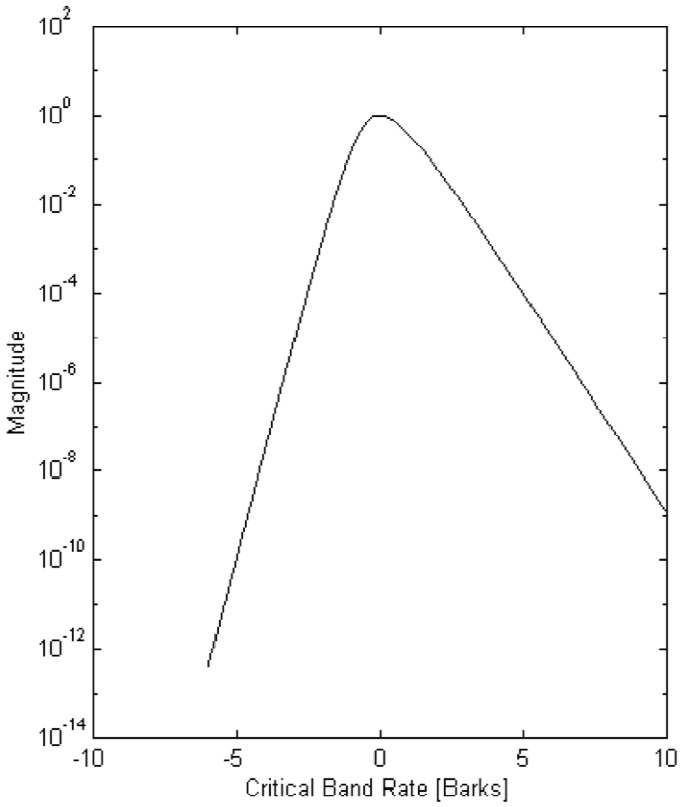


Figure 4-6: Model of Spreading Function B(z) plotted against normalized Bark scale

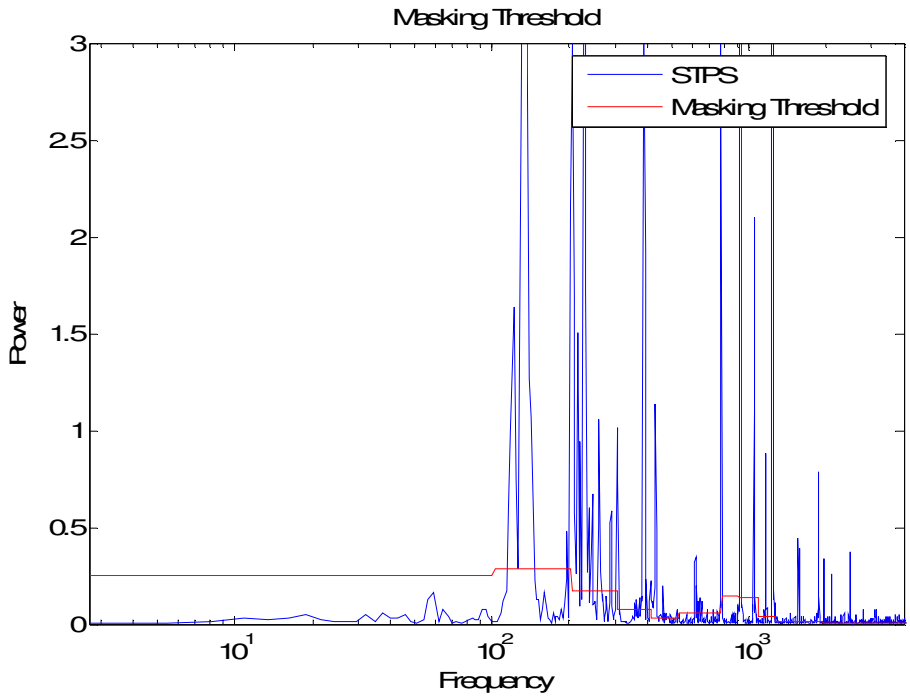


Figure 4-7: Plot of STPS and the Masking Threshold

4.1.3 Suppression Function

The goal of the suppression function is to reduce the noise components to a level where they become inaudible. The masking threshold establishes this level. However, the masking threshold is determined by the STPS $Y_p(f)$ and if the power spectrum changes the masking threshold changes as well [1]. This leads to an iterative approach to attenuating noise components [1]. The decision rule for each iteration step k is defined as

$$\left|\hat{S}^k(f)\right| = \begin{cases} \left|Y^k(f)\right|, & \left|Y^k(f)\right|^2 \geq T(f) \quad [\text{harmonic signal is present}] \\ & or \\ & \left|Y^k(f)\right|^2 < M^k(f) \quad [\text{noise is masked}] \\ H^k(f)\left|Y^k(f)\right|, & \left|Y^k(f)\right|^2 < T(f) \quad [\text{noise is present}] \\ & and \\ & \left|Y^k(f)\right|^2 \geq M^k(f) \quad [\text{noise is not masked}] \end{cases} \qquad [1]$$

where the noise suppression filter function is defined as

$$H^k(f) = \frac{\sqrt{M^k(f)}}{|Y^k(f)|} \qquad [1]$$

During each step the noise components are attenuated to approximately the masking threshold M(f). The STPS is then updated with the noise components attenuated. This updated STPS is then used to calculate a new masking threshold M(f) which is applied to the STPS attenuating the noise components even more to the new level of the masking threshold. Each segment of audio being filtered underwent two iterations of suppression for the components determined to be noise as specified in [1].

4.2 Noise Suppression Filter Sound Fidelity Improvements

The focus of this thesis is to investigate possible improvements to the sound quality of the filtered signal. The thresholding technique described above determines what frequency bins are desired signal and which are not. Often, the resulting noise suppression filter will remove higher harmonics of the desired signals or percussion sounds because the spectral magnitudes of these harmonic peaks fall below the decision threshold. This leads to a reduction in the noise but also a less faithful representation of the original signal. The listener's perception is that less noise is present, but also that the sound quality is diminished, particularly in terms of high frequency fidelity (audio "brightness"). Therefore, three different techniques were investigated for this research project in an attempt to maintain the efficacy of the noise reduction while mitigating the

loss of fidelity in the music. The improvements investigated were prediction and preservation of signal harmonics, synchronizing noise filter revision to the song tempo and primary beat patterns, and noise filter relaxing for detected percussion events.

4.2.1 Preservation of Signal Harmonics

When a note is played by a musical instrument, the frequency spectrum of the sound is made up of a fundamental frequency and numerous harmonics that occur at integer multiples of the fundamental frequency. Thus it is straightforward to determine the harmonic frequencies if the fundamental pitch is known. A peak detector can be used to determine which frequencies include both fundamental and harmonic peaks in the magnitude spectrum after the signal is passed through the initial spectral signal detection threshold. Narrow pass bands are then opened up in the noise suppression filter spectrum at multiples of these harmonic peak frequencies. These added pass bands allow more of the musical content to be preserved during noise suppression. Of course, inclusion of too many additional harmonically spaced filter pass bands will result in more high frequency noise passing as well.

4.2.1.1 Harmonic Prediction Implementation

The harmonic prediction algorithm simply uses a peak detector to determine which frequencies have peaks in the magnitude spectrum after noise suppression filtering. A loop then calculates a user-defined number of harmonics of each potential fundamental frequency the peak detector found. A narrow pass band of user-defined width is then placed in the filter spectrum centered at each of the harmonic frequencies. The following

figures show the filtered spectrum with and without the harmonic prediction function. Figure 4-6 clearly shows that there are four peaks that are not included in the filtered spectrum because they do not possess enough power to exceed the signal detection threshold. This leads to a decrease in sound quality as the noise suppression filter removes these harmonics. In Figure 4-7, pass bands are placed around the harmonics in the noise suppression filter, and so the signals in these bands are included in the filtered spectrum. This leads to improved fidelity.

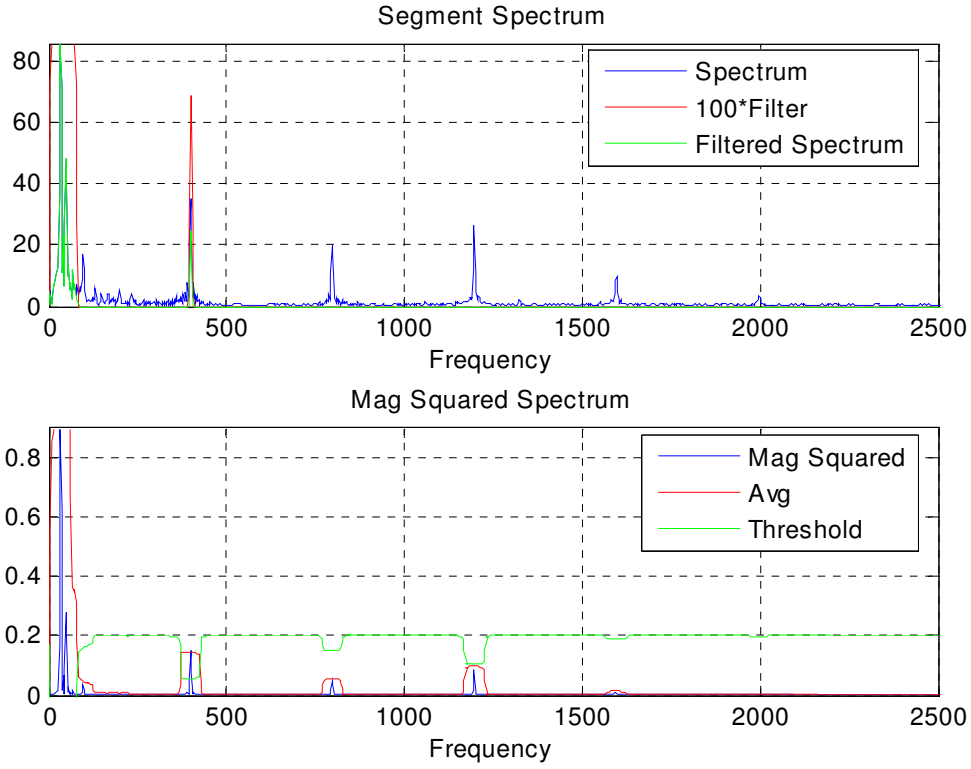


Figure 4-8: Top showing the original spectrum, the filter response, and the filtered spectrum without harmonic prediction. Bottom showing the power spectrum and the different components of the threshold.

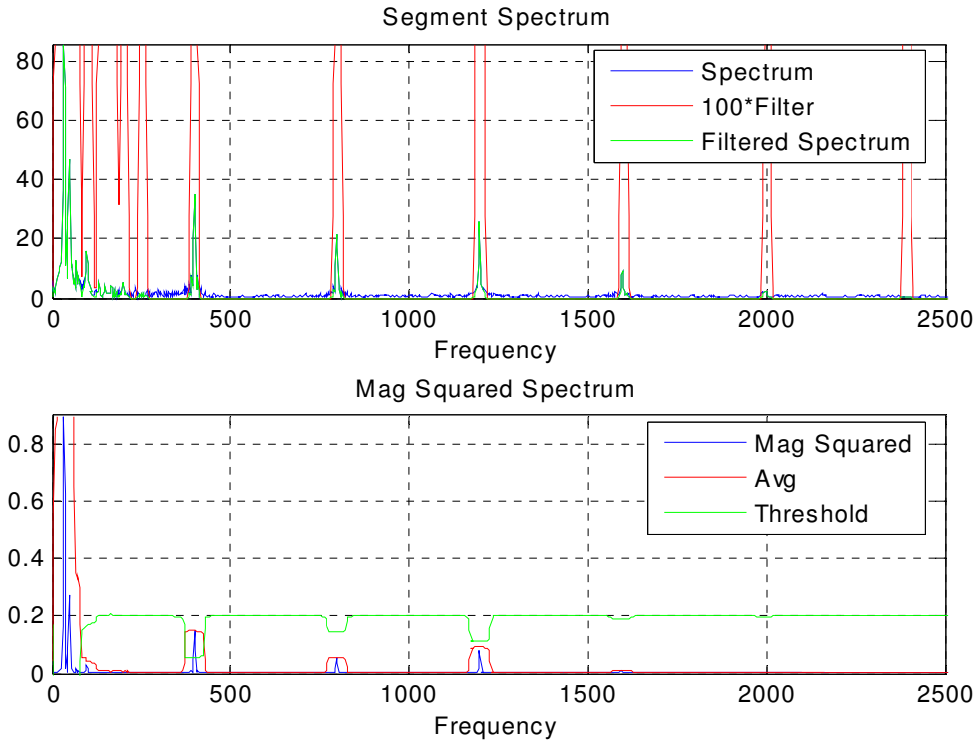


Figure 4-9: Top showing the original spectrum, filter response, and filtered spectrum with harmonic prediction. Bottom showing the power spectrum and the components that make up the threshold.

4.2.2 Song Tempo Synchronization Processing

The filter uses a fixed window length for segmenting and processing the incoming audio signal. The length of the window can be optimized for the song being filtered. The window length is important as it affects the accuracy of the frequency spectrum used to determine the threshold. A longer window length allows for better spectral resolution since more samples results in closer spacing of the frequency bins of the FFT. This allows frequencies that are close together to be better resolved. However, as the window length is increased and the spectral resolution increases, the time resolution is decreased as each processing segment covers a longer time period. When altering the frequency spectrum of an audio signal by manipulating its magnitude spectrum alone, it is important

to localize only one musical event in each window since any temporal information is effectively lost after taking the magnitude of the FFT. By selecting window lengths that correspond to a particular subdivision of the music, *i.e.* a quarter note, eighth note *etc*., only one musical event will be captured in each window.

4.2.3 Percussion Preservation (drums, cymbals, etc.)

Percussion instruments also pose a problem for the simple threshold-based noise filtering technique. The frequency spectrum of a drum hit looks very similar to noise – broadband spectral content with low amplitudes. Therefore, the threshold technique treats percussion hits as noise elements, and suppresses most of the frequency content of percussion. This leads to a muffled, low-pass filtered sound for most percussion instruments, which produce non-harmonic, broadband, impulsive sounds. By determining when drum hits occur in the audio signal, it is possible to improve the perceived sound quality by relaxing the noise suppression filtering at those samples, allowing the spectral content of the percussion to be passed through.

4.2.3.1 Beat Tracking

In order to relax the noise suppression filter for percussion and drum hits, the samples corresponding to these hits must be determined ahead of time. This necessitates a beat time tracking (tempo tracking) algorithm. Estimating the tempo of a piece of music is a complex problem consisting of two objectives, estimating the number of beats per minute (bpm) at which the music is played and determining exactly when the beats occur [8]. The algorithm used here is based on both [8] and [9].

The first step is to calculate the energy flux, which also called the onset envelope in this context. This requires locating large changes in frequency content occurring across multiple frequency bands. These large changes in frequency content are usually associated with note onsets and percussion hits. Short time Fourier transforms (STFT) are employed for this calculation. Short segments of the signal are extracted at regular instants, multiplied by an analysis window, and transformed into the frequency domain [8]. Denoting x[n] as the signal, t_i the frame time in seconds, F_s the sampling frequency, and N the size in samples of the analysis window h[n], the short time Fourier transform $X(f,t_i)$ at the normalized frequency f and frame is

$$X\left(f,t_i\right)=\sum_{n=0}^{N-1}h[n]x\left[n+F_s t_i\right]e^{-j2\pi fn} \qquad [8]$$

The window size used was 10 milliseconds, with a 10 millisecond window advance (no overlap) between frames as suggested by [8]. The following steps only use the magnitude of the STFT. Each STFT is multiplied by a nonlinear monotonic compression function $G\left(\left|X\left(f,t_i\right)\right|\right)$ so that high frequency content such as high hat cymbal hits are not masked by high amplitude low-frequency components such as bass notes [8]. The compression function used was $G\left(\left|X\left(f,t_i\right)\right|\right)=\left|X\left(f,t_i\right)\right|^{1/2}$. The first order difference between compressed STFT results is taken from frame to frame, and the result is summed across all frequency bins [8]. This result is then half wave rectified to obtain a positive energy flux signal, which exhibits sharp maxima at transients and note onsets. This signal can be shown as

$$\hat{E}(i)=\sum_{f=f_{\min}}^{f_{\max}}G\left(\left|X\left(f,t_i\right)\right|\right)-G\left(\left|X\left(f,t_{i-1}\right)\right|\right) \qquad [8]$$

$$E(i) = \begin{cases} \hat{E}(i) & \hat{E}(i) > 0 \\ 0 & otherwise \end{cases} \qquad [8]$$

where f_{min} and f_{max} are the range of frequencies over which the summation is carried out [8]. These were set to 100 Hz and 10 kHz respectively as recommended by [8].

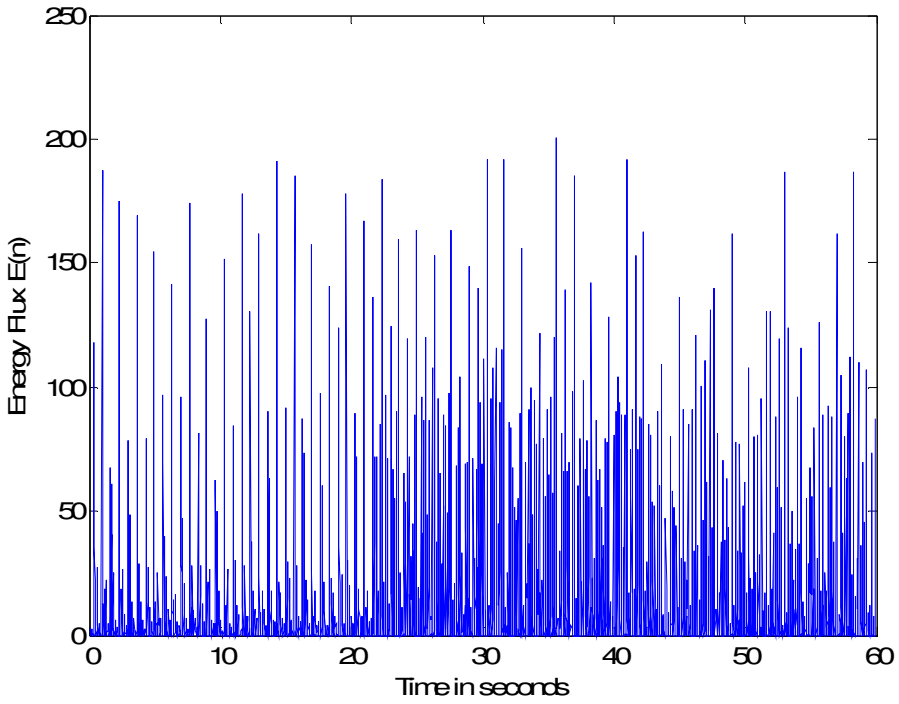


Figure 4-10: The Energy flux for a rock song.

In order to determine the tempo of the song, the calculated energy flux is then compared to an expected energy flux. In the algorithm used in this thesis, multiple expected energy flux signals are created corresponding to tempos of 60bpm up to 150bpm. The expected energy flux signal is modeled simply as a series of discrete pulses.

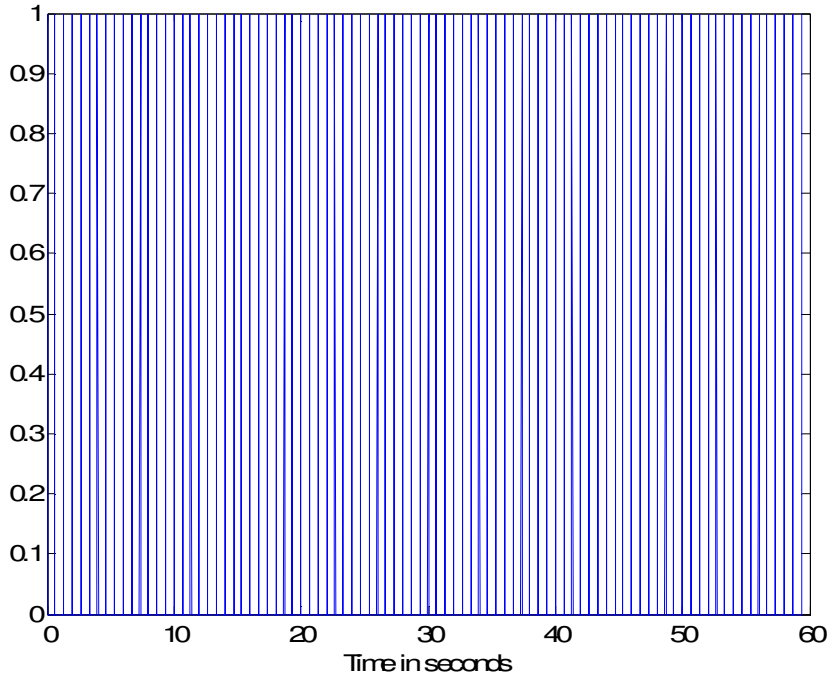


Figure 4-11: Impulse signal representing tempo of 90bpm.

The expected energy flux signals are then correlated with the calculated energy flux signal. The correlation lag time with the largest peak is chosen as the tempo period for the song. A dynamic program that is detailed in the next section and in [9] then take the calculated energy flux signal and the tempo period as inputs and returns the indices that correspond to the optimum set of beat times.

4.2.3.2 Determining beat locations

Now that the tempo and therefore beat period is known, the actual beat locations (time indices) can be determined. The objective is to determine a sequence of beat times that both correspond to maxima in the onset strength envelope and that also have a regular spacing that corresponds to the calculated tempo. An objective function is defined in [9] that accomplishes these goals simultaneously, and is given as

$$C(\{t_i\}) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{i=2}^{N} F(t_i - t_{i-1}, \tau_p)$$ [9]

where $\{t_i\}$ is a sequence of N beat instants. $O(t)$ is the onset strength envelope calculated from the audio, α is a weighing factor to balance the importance of the two terms, and $F(\Delta t, \tau_p)$ is a function that measures the consistency between an inter-beat spacing Δt and the ideal beat spacing, τ_p [9]. In this case the function F is defined as

$$F(\Delta t, \tau) = -\left(\log \frac{\Delta t}{\tau}\right)^2$$ [9]

which takes on a maximum value of zero when $\Delta t = \tau$, and becomes increasingly negative for larger deviations [9].

The score C is then assembled recursively by calculating the best score for all possible sequences ending at time t by way of

$$C(t) = O(t) + \max_{\tau=0\ldots t} \{\alpha F(t - \tau, \tau_p) + C(\tau)\}$$ [9]

This equation states that the best score for time t is the local onset strength plus the best score to the preceding beat time τ that maximizes the sum of that best score and the transition cost from that time [9]. The actual preceding beat time that gave the best score must also be stored and is given by

$$P(t) = \arg\max_{\tau=0\ldots t} \{\alpha F(t - \tau, \tau_p) + C(\tau)\}$$ [9]

In this case we only search a limited range of τ since the term F will make it unlikely that the best predecessor lies far away from t-τ_p [9]. The range that is searched is from τ=t-$2\tau_p \ldots t$-$\tau_p/2$ [9]. C and P are calculated for all times and the maximum of C is the last beat time. We then work backwards via P to find the preceding beat times. These beat times are given in samples.

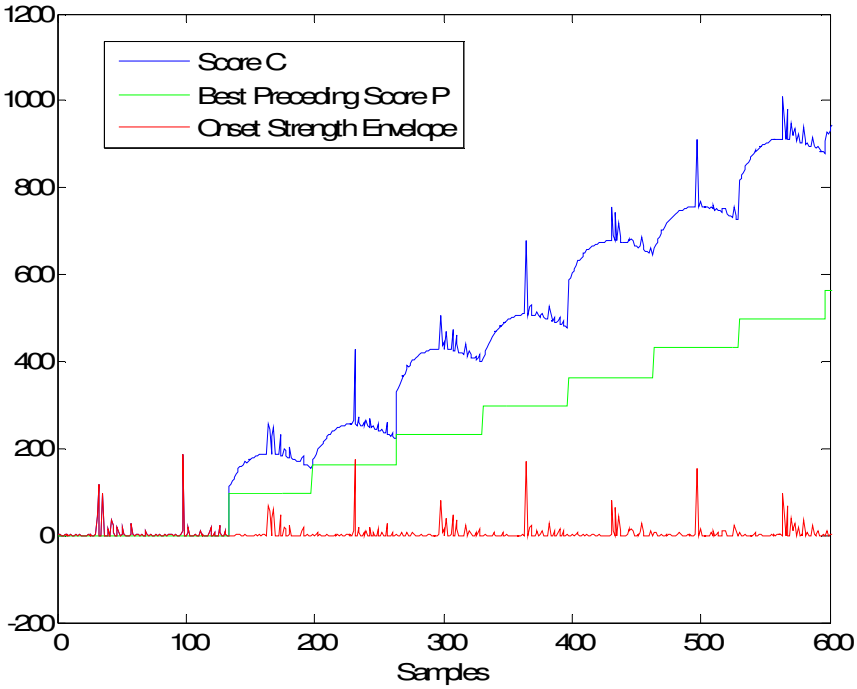


Figure 4-12: Plot showing the different pieces of the recursive scoring equation discussed above

Figure 4-10 shows the different components of the equations used to determine the actual beat locations. The onset strength envelope is shown in red with the recursive score C shown in blue. The green graph is the best preceding beat location. The amplitude of the green graph P corresponds to the sample index of the best preceding beat location.

4.2.3.3 Passing segments that contain beats

Now that the sample indices corresponding to when a beat occurs are known the filter can be relaxed at these samples to avoid strong filtering of percussion hits. One problem with filtering the percussion in this fashion is that often times the window boundaries of the signal don't line up exactly with the beat locations. Once a window is transformed into the frequency domain time resolution in that frame is gone. Therefore if a beat falls somewhere inside a window the filter must be relaxed for the entire window

in order to not filter the beat. This leads to noise possibly being passed before the beat takes place or the premature filtering of the decaying beat because it falls in the next window. In order to solve this problem, when a processing data window contains a beat the starting point of the window is realigned to correspond with the beat location. This allows for more control in relaxing the filter only for the duration of beat; and ensures that the noise filter is not opened before the percussion hit begins. This also leads to fractions of the original window being "leftover". These "leftover" fractions of the original window are filtered using the filter calculated for the previous, pre-beat window. Care must be taken to keep track of the lengths and locations of the "leftover" portions of windows that result from this realignment so they are correctly overlapped and added when the signal is being reconstructed after filtering. The window re-alignment is shown in the following figure. In figure 4-10 the arrows at the top represent the locations of the beats. The numbers at the top are the window numbers. The green bars represent the starting point of each window. The brackets with the numbers below them represent which window the filter was derived from. Filter 2 (F2) was derived from window 2; Filter 3 (F3) was derived from window 3 and so on. The "leftover" pieces due to the re-alignment use the filter derived from the preceding window. The windows with no brace are the beat windows, which are passed by the filter.

Choosing a window length based on a sub division of the beat period so that the beat onset always falls at the beginning of the window may have been an easier implementation. However that would constrict the window length, which may or may not be optimal for filtering.

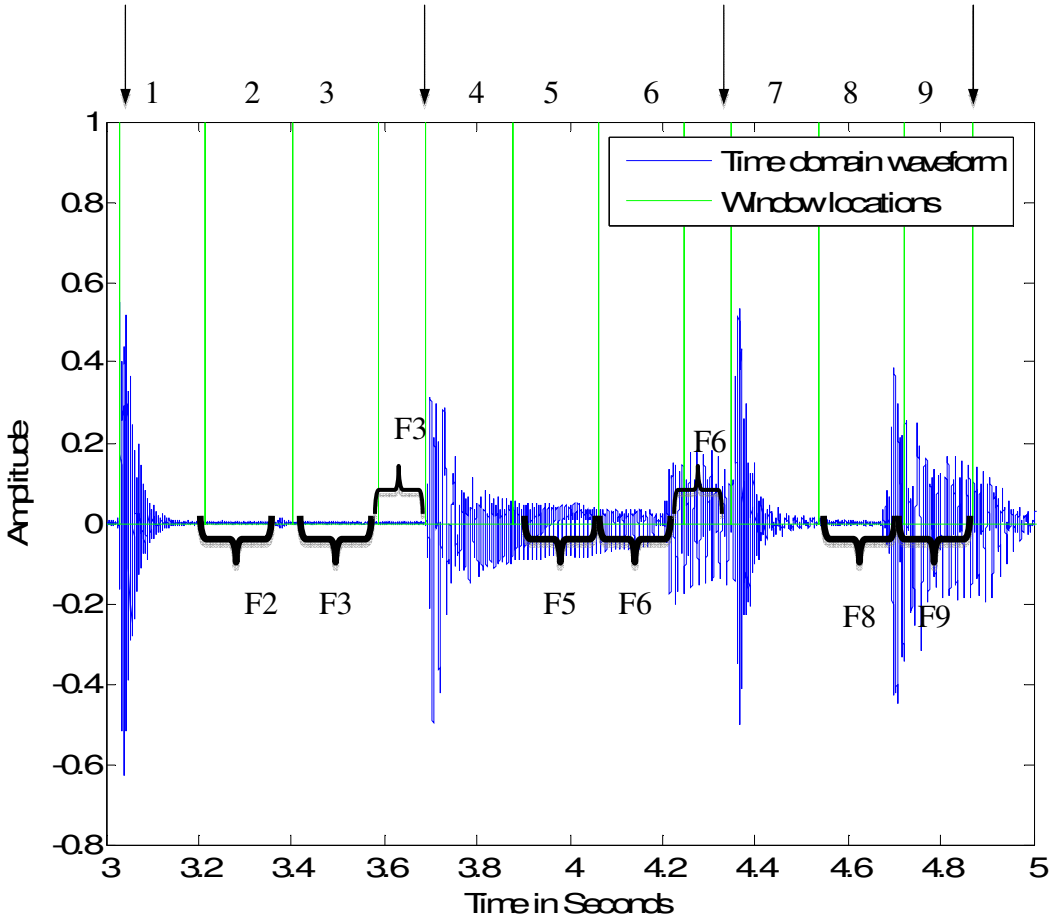


Figure 4-13: Showing window re-alignment on beats.

5 Test & Results

Three different metrics were used to quantify the improvements made to the noise-filtering algorithm. These three metrics were signal to noise ratio to determine the amount of noise reduction, percent difference as a measure of signal distortion, and "audio quality" as determined by a subjective listening test. These metrics were applied to several different musical audio excerpts that were chosen to isolate particular performance aspects of the noise filtering.

To characterize the effectiveness of the filter in reducing broadband random noise, a clean audio signal was corrupted with 30 dB of white Gaussian random noise and processed through two noise filters – one that included the improvement features of interest, and one that did not. The clean audio signal was also processed through each filter in order to determine how much the filter introduces unwanted distortion of the original audio signal, generally by removing desired signal content. Any deviations in the harmonic shape of the filtered spectrum compared to the original uncorrupted signal spectrum are interpreted as "noise" in the SNR computations, and are similarly detected in the percent difference distortion measurements. Ultimately the most important criteria for determining if there was any improvement between the original algorithm and the algorithm with the previously described additions is the subjective listening test. If listeners rate the filter with feature additions higher than the original filter, these would constitute positive results since human perception is ultimately what matters most.

5.1 SNR Comparison

Signal to noise ratio is used to quantify how well the filter attenuates the broadband noise. In this study, the signal to noise ratio is computed separately for the filter input signal and the filter output signal. These computations are repeated for each audio segment processing window, and average values are computed for the total audio signal excerpt. The difference between the filter input SNR and the filter output SNR, when expressed in decibels, quantifies the noise reduction (for positive SNR changes) or signal degradation (for negative SNR changes) introduced by the noise filter.

The filter input signal SNR requires estimates of the noise power and signal power of the noise-corrupted audio test signals. For the noise-corrupted filter input signal, the added Gaussian noise is zero mean and uncorrelated. Therefore, the noise magnitude spectrum can be simply obtained by subtracting the original uncorrupted signal magnitude spectrum from the magnitude spectrum of the signal after noise addition. This noise magnitude spectrum is then squared and summed across all frequency bins, resulting in the total input noise power for the current window segment of the test signal. The input "signal" power for that window is then determined simply by summing up the power of the original uncorrupted signal spectrum. Finally, the filter input signal to noise ratio (SNR_{in}) for each window is computed by taking the ratio of the uncorrupted signal power with the noise power found in the corrupted signal. The SNR's for each window are averaged and converted to decibels, giving an overall SNR for the test case. The input test signal overall SNR_{in} is given then by the following expression:

$$SNR_{in} = 10\log_{10}\frac{1}{M}\sum_{n=0}^{M-1}\left(\frac{\sum_{k=0}^{N-1}\left|S(k)\right|^2}{\sum_{k=0}^{N-1}\left\|Z(k)\right|-\left|S(k)\right\|^2}\right) \quad (5.1)$$

where $S(k)$ is the sampled uncorrupted signal spectrum, $Z(k)$ is the sampled noise corrupted spectrum, k is the frequency sample index (for N frequency sample bins), and n is the temporal window number (for M windows).

The output signal to noise ratio (SNR_{out}) is calculated in a similar way, except that the filtered output spectrum replaces the corrupted input signal spectrum in the noise power computation. The difference between the filtered signal magnitude spectrum and the original, uncorrupted signal magnitude spectrum is calculated leaving the residual noise magnitude spectrum of the filter output for each window. This noise spectrum is then summed across all frequency bins resulting in the output noise power. The "signal" power for the filter output SNR calculation is again taken to be the total power in the uncorrupted signal spectrum. An overall output SNR is again determined for each test case by averaging the values across all sample windows and converting the aggregated result to decibels.

The filter output SNR_{out} is then given by:

$$SNR_{out} = 10\log_{10}\frac{1}{M}\sum_{n=0}^{M-1}\left(\frac{\sum_{k=0}^{N-1}\left|S(k)\right|^2}{\sum_{k=0}^{N-1}\left\|f(k)\right|-\left|S(k)\right\|^2}\right) \quad (5.2)$$

where $f(k)$ is the sampled filter output spectrum (for N frequency samples).

5.1.1 SNR Comparison of Solo Piano

A solo piano audio excerpt without percussion was chosen for one SNR comparison in order to accurately isolate and measure the amount of broadband noise reduction

achieved by each version of the noise filter. For example, a reduction in the filter output signal SNR when adding the harmonic prediction feature to the noise filter would indicate that a measureable amount of additional noise was allowed to pass through the filter as signal harmonic passbands are opened. Conversely, harmonic prediction should preserve more of the true signal spectrum content, and therefore it is expected that the noise filter with harmonic prediction should produce a higher output signal SNR than the filter without harmonic prediction, as any signal harmonic content removed by the filter would appear in the noise term of the SNR calculation. This is because the output signal SNR calculation treats all spectrum changes (more noise or less signal) between the uncorrupted signal and the filtered signal (with less noise and less percussion) as part of the noise power.

Solo piano is a purely harmonic instrument, so that its frequency spectrum does not contain the broadband, noise-like appearance that is characteristic of non-harmonic percussion instruments. Percussion events in the SNR test signal have the effect of lowering the calculated filter output SNR because the filter cannot distinguish percussion signal from noise, and therefore tries to filter out some of the percussion. This will cause significant differences in spectral magnitude values for frequency bins that contain percussion in the original signal compared to the filtered output signal. Thus, any removal of percussion by the filter will be interpreted in the SNR calculation as an increase in noise and will degrade the SNR value. Therefore, any desirable reductions in the added broadband random noise by the filter could be offset in the SNR calculation by any unwanted reductions in the percussion signal. By calculating the SNR on an audio

signal that does not contain any percussion, the filter's ability to reduce undesired broadband random noise can be isolated and quantified.

Using Matlab's additive white Gaussian noise function, 30dB of random noise was added to a 30 second solo piano audio excerpt. The input and output SNR's were calculated for both the original threshold-based noise filter and the filter with harmonic prediction. The results, summarized in Table 5-1, show that for purely harmonic instruments (*i.e.* no percussion) the harmonic prediction adds an additional **1.5** dB of SNR improvement over the original algorithm. This results appears to support the conclusion that the harmonic prediction feature is providing more beneficial effect restoring true harmonic signal content than it is losing by allowing a little more noise through the filter in the harmonic signal passbands.

Regular Thresholding Filter	Filter with Harmonic Prediction
SNR_{in} = 32.20 dB SNR_{out} = 35.96 dB	SNR_{in} = 32.21dB SNR_{out} = 37.51dB
ΔSNR = 3.76 dB	ΔSNR = 5.30 dB

Table 5-1: Signal to noise ratio comparison for both the original filter and the filter with harmonic prediction

5.1.2 SNR Comparison of Drum Solo

A drum solo test signal was also used to quantify the SNR performance for percussion instruments passages. Since the magnitude spectrum of drums resembles that of broadband noise, we would expect the noise filter to perform poorly with this test signal, since it will have a difficult time distinguishing the percussion signal spectrum from the added noise in the spectrum. This filtering out of the desired percussion signal by the noise filter leads to a muffled, low-pass filtered sound in the drum solo test signal, with

an almost complete loss of any cymbals or hi-hat sounds. The calculated filter output SNR may even be lower than the input signal SNR, implying that there is more noise (actually less desired signal) in the filtered output signal than in the noise corrupted input signal. This effect should be diminished by the added feature that allows the noise filter to pass information that falls on or just after the tempo beats in the song. Since the percussion is generally in charge of maintaining the tempo in a song, it is expected that most of the drum hits will fall on the beat or some subdivision of the beat duration. If so, then an increase the output SNR of the drum solo excerpt should be observed when this feature is added to the noise filter.

As with the harmonic piano excerpt, 30dB of additive white Gaussian noise was added to the drum solo excerpt using Matlab's `awgn( )` function. The input and output SNR was calculated for both the original thresholding noise filter and the filter with the feature added to allow passing of percussion on the tempo beats. The results of this assessment are shown in Table 5-2. There is significant improvement (+8 dB change) in the output signal SNR for the drum solo excerpt when passing signals on and shortly after the tempo beats compared with the original noise filter. In all cases, however, as expected the filter output SNR for the percussion excerpt is lower than the SNR for the noise-corrupted filter input signal.

Regular Filter	**Filter that passes the beats**
SNR_{in} = 32.41 dB SNR_{out} =11.59 dB	SNR_{in} = 32.42 SNR_{out} = 19.95
ΔSNR = -20.82 dB	ΔSNR = -12.47 dB

Table 5-2: Signal to noise ratio comparison of both the original filter and the filter that passes the beats

5.1.3 SNR Comparison of Rock Song

A rock song is generally composed of multiple harmonic instruments (guitars, keyboards), voices (also harmonic), and percussion. For this sort of audio except, the noise filter should be able to effectively distinguish the harmonic content (*i.e.* the harmonic instruments and voices) from the noise, producing a strong filter output SNR for passages dominated by harmonic content. However, as was discussed earlier, filtering percussion excerpts should prove to be problematic, leading to degradation in the overall filter output SNR calculation for passages with mostly percussion. The rock song excerpt combines these two phenomena together in the same audio test track; demonstrating the noise filter's performance in a fully integrated and typical set of test conditions.

As with the solo piano and drum solo excerpts, 30dB of white Gaussian noise (AWGN) was added to the clean rock song signal excerpt and then applied to the noise filter. The input and output SNR's were calculated for the original filter and all different combination of filter improvements. The results are given in Table 5-3.

Filter Type	Input SNR	Output SNR
Original Filter	32.88 dB	22.87 dB
Filter with harmonic prediction	32.88 dB	24.84 dB

Filter passing beats	32.88 dB	28.90 dB
Filter with harmonic prediction and passing beats	32.88 dB	28.90 dB

Table 5-3: Comparison of signal to noise ratios for different filters using a rock song excerpt

Filtering the solo piano showed that the filter does attenuate noise with minimal loss of harmonic signals. However the filtered drum signal's output SNR was much lower than the input SNR because much of the original percussion signal gets filtered, as it is considered to be the same as noise by the filter. The conclusion from the above results is that since the output SNR is lower than the input SNR for a rock song, the SNR performance is dominated by the attenuation of the drum components rather than lack of attenuation of the broadband noise. This will be corroborated with the subjective listening test, since human perception is ultimately the most important criterion.

Another test conducted with a rock song was calculating output SNR if the filter threshold is only re-estimated one window after a beat. The thinking behind this is that the harmonic structure of a song tends to change on the beats or subdivisions of the beat. Re-estimating the noise filter spectrum only at certain times rather than for every processing window would greatly reduce the time it takes to filter a whole song. Table 5-4 shows that there is very little difference in output SNR for this case versus re-estimating the filter threshold every window.

Filter Type	Input SNR	Output SNR
Filter passing beats – updated every window	32.88 dB	28.90 dB
Filter passing beats – updated 1	32.88 dB	28.72 dB

window after beat		
Filter with harmonic prediction and passing beats – updated every window	32.88 dB	28.90 dB
Filter with harmonic prediction & passing beats – updated 1 window after beat	32.88 dB	28.90 dB

Table 5-4: Comparison of signal to noise ratios for different filters updated one window after the beat vs. updating every window.

5.2 Distortion Comparison

Measures of distortion are common in audio applications. Total Harmonic Distortion (THD) is one such common metric. THD is the ratio of the sum of the powers of all the harmonics of a signal to the power of the fundamental. In an ideal audio amplifier the spectrum of the output signal should be identical to the spectrum of the input signal. In real audio amplifiers non-idealities and non-linearity's add harmonics to the signal, which is considered harmonic distortion.

For this study, a percent difference calculation is being used to quantify how different the filtered spectrum of a clean audio signal is from the original signal spectrum, to determine the degree to which the filtered spectral shape is being distorted by the noise filter. This is computed similarly to the output SNR values, except that the differences are computed on the magnitude spectra rather than the power spectra, and the results are expressed on a linear scale as a percentage rather than a logarithmic dB scale. In order to calculate the percent difference, the clean original signal is passed through the filter without adding any noise and the spectral content of the input and output are compared. The magnitude spectrum of the original clean input signal is subtracted from the magnitude spectrum of the filtered signal output for each window, resulting in the difference magnitude spectrum. This difference magnitude spectrum is then summed

across all frequency bins. The clean magnitude spectrum for each window is also summed across all frequency bins. The ratio between the difference magnitude spectrum sum and the clean magnitude spectrum sum is computed; yielding the percent spectral difference for each window. The average of all the percent differences across all processing windows is found, resulting in a single, overall percent difference value. The percent difference is given then by

$$\%diff = 100 * \frac{1}{M} \sum_{n=0}^{M-1} \left(\frac{\sum_{k=0}^{N-1} \left\| f(k) \right| - \left| S(k) \right\|}{\sum_{k=0}^{N-1} \left| S(k) \right|} \right) \quad (5.3)$$

where *f* is the filtered signal spectrum, *S* is the clean signal spectrum, k is the sample index, and n is the window index from 1 to M.

5.2.1 Distortion Comparison of Solo Piano

As discussed in section 5.1.1, a solo piano was used to quantify the noise reduction effectiveness of the filter. Since a piano has a harmonic spectrum the filter should easily separate the noise from the signal and filter out the noise accordingly. This also means that the filter should leave the harmonic content unchanged resulting in very little distortion. This is reflected in the results with the original filter giving a distortion of between 1.5 and 2 percent and the filter with harmonic prediction coming in at between 1 and 1.5 percent. In conjunction with the results in section 5.1.1 this shows that the filter removes noise with minimal distortion to signal as long as the signal is comprised of harmonic instruments or voice. The addition of harmonic prediction improves SNR while also lowering distortion.

In the case of a purely harmonic signal there was a decrease of 0.34 % in distortion when compared to the original algorithm.

Regular Filter	Filter with Harmonic Prediction
% diff = 1.74 %	% diff =1.40 %

Table 5-5: Comparison of distortion between regular filter and filter with harmonic prediction for a solo piano (percussion-free) excerpt

5.2.2 Distortion Comparison of Solo Drums

Drums have a spectrum similar to that of noise so it is difficult for the filter to separate noise from desired signal in percussion passages. This leads to filtering out spectral components of the percussion signal that should be passed by the filter; even when filtering an excerpt that has not been corrupted by noise. The noise filtering will cause large differences in the spectrum between the original signal and the filtered signal even though there was never any noise present in the signal being passed through the filter.

This large amount of distortion is obviously undesirable since it means a loss in sound quality for percussion instruments. The addition of the noise filter feature allowing the filter to pass signals when beats occur drastically reduces the distortion the signal suffers when compared to the original filter.

In the case where a signal consisting of purely percussion was used, there was a decrease of 14.52% in distortion when compared to the original algorithm, as shown in Table 5-6.

Regular Filter	Filter that passes the signal on the beat
%diff = 28.96 %	% diff = 14.44 %

Table 5-6: Comparison of distortion between regular filter and filter that passes the signal in the beat, for a drum solo percussion excerpt

5.2.3 Distortion Comparison of a Rock Song

Since a rock song consists of many different instruments the percent difference between the original signal spectrum and the filtered signal spectrum will fall somewhere in between that of the solo piano and the solo drums. The additions of harmonic prediction and allowing the filter to pass the signal on the beats decrease the distortion when compared to the original algorithm. The greatest decrease in distortion happens when the filter passes the beats, allowing more of the percussion signals to pass through the filter. Interestingly when the filter is changed to re-estimate the threshold one window after a beat occurs rather than every window the distortion drops even further. As with the solo piano and drums, a rock song was filtered without any noise added to it. The distortions for various combinations of filtering are shown in the following table.

Filter Type	Percent Difference
Regular Filter	17.83 %
Filter with harmonic prediction	15.23 %
Filter passing signal on beats	7.18 %
Filter passing signal on beats and with harmonic prediction	7.15 %
Filter passing signal on beats and re-estimating threshold one window after beat	5.84 %
Filter passing signal on beats with harmonic prediction and re-estimating threshold one window after beat	5.78 %

Table 5-7: Comparison of distortion between different filters

5.3 Effect of Impulsive noise on Beat Tracking Algorithm

Impulsive noise is noise of a very short duration such as a pop or click that one may hear when listening to a vinyl LP due to a scratch across the grooves. In this particular case the popping sound will be periodic as the scratch passes under the needle with each revolution of the LP. This could potentially fool the beat tracking algorithm into choosing the tempo associated with the periodic popping as the tempo for the song rather than the songs true tempo. This would lead to incorrect beat locations that would render the percussion preservation features ineffective.

Tests of the beat tracking algorithm were conducted by adding different periodic clicks to samples of different styles of music. The clicks added corresponded to repetition frequencies that were different from the true tempo of the song. For electronic style songs with a very strong defined beat the beat tracking was not thrown off by the impulsive noise and correctly calculated the true tempo of the song. This same test was also conducted using the same rock song used in the previous noise and distortion tests. When the impulsive noise had a large amplitude, the beat tracking algorithm was fooled

and chose the impulsive noise as the tempo of the song. When the amplitude of the impulsive noise was reduced, the beat tracking algorithm chose the true song tempo.

Another factor in this effect is the duration of the impulsive noise. If the periodic popping only lasts for seconds and the song length is several minutes long, the algorithm chooses the true song tempo. We can conclude that the beat tracking can be fooled by impulsive noise. However this is dependent on many factors including the strength of the impulsive noise, the strength of the instrumentation that define the true song beat, and the amount of time the periodic impulsive noise is heard in the song.

5.4 Filtering Authentic Noisy Source Material

A brief test of the noise reduction filters was conducted using an existing excerpt of audio source material that was corrupted by noise from the actual processes of broadcasting and recording, rather than by artificially adding noise of a particular strength and bandwidth. The song used was an FM broadcast of a live concert recorded to tape. This particular recording exhibits tape hiss (broadband noise) along with other noise due to the live recording environment.

This recording was filtered using the original broadband noise filter and the spectrum of the original signal was compared with the spectrum of the filtered signal. Based on the following figures it can be shown that the noise floor was lowered by the filter which corresponds to attenuation of noise components which is a desirable result.

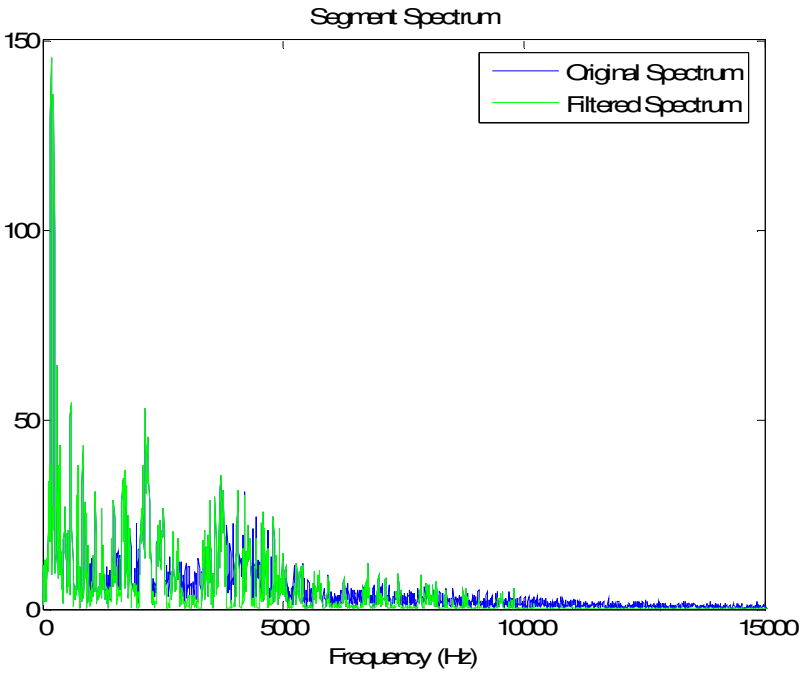


Figure 5-1: Frequency Spectrum of one segment showing the original spectrum (blue) and the filtered spectrum (green)

As can be seen in figure 5-1, the noise components above 10kHz are all attenuated.

Many of the noise components between 5kHz and 10kHz are also attenuated.

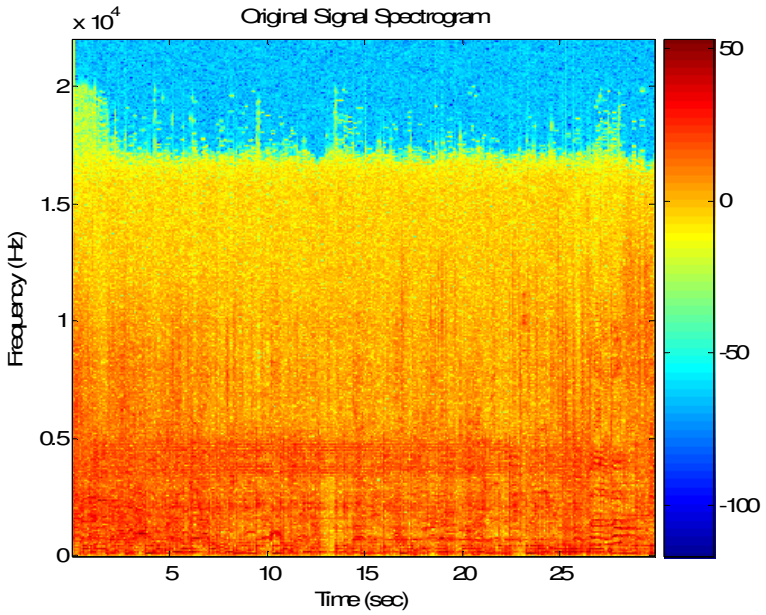


Figure 5-2: Spectrogram of noisy song.

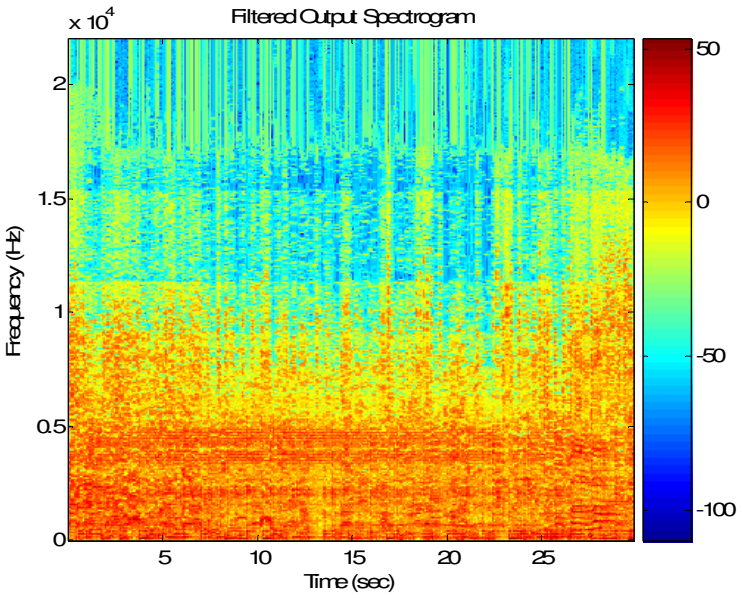


Figure 5-3: Spectrogram of Filtered Song

Figures 5-2 and 5-3 show spectrograms of both the original signal and the filtered signal. A spectrogram is a method of showing both time and frequency. Comparing figures 5-2 and 5-3, there is a noticeable attenuation in the components from about 10kHz up to between 15kHz and 20kHz. There is also some attenuation of components between 5kHz and 10kHz. With most of the audio frequency components below 5kHz, the filter has attenuated most of the noise components while leaving the signal components intact.

Upon listening to the filtered audio it is obviously apparent that the broadband noise has been almost completely eliminated. There is still some tonal noise present due to the live set up and associated electrical hum. The addition of harmonic prediction does help to restore some of the "crispness" of the original recording without any noticeable difference in noise level. As would be expected the percussion suffers from filtering however this recording is of low audio quality to begin with so there isn't much difference in quality between the filtered and unfiltered percussion.

5.5 Subjective Listening Test

The subjective listening test involved listening to a series of composite audio files. The format for these files is, 10 seconds of audio signal plus noise, followed by 10 seconds with one filter implementation, then 10 more seconds another filter implementation. The listeners were told of the file format, but nothing about which filters were employed for each segment. The listeners were then asked to assess the audio quality of each segment on a scale from 0 to 5; with 0 being unbearable audio quality, and 5 being CD quality.

Five anonymous, untrained people with no experience in listening for particular traits or anomalies in music were chosen to listen to each file one at a time in a quiet room with a pair of Sony noise cancelling headphones. The noise-cancelling feature was turned off for the tests. The particular headphones were chosen for their high quality sound. Three tests were conducted with various filter techniques represented and two different additive noise levels. The results below show that the filter with added improvements outperformed the original filter in every subjective test. Filter 1 is the original filter and Filter 2 is the improved filter with harmonic prediction and allowing the filter to pass percussion on the tempo beats. Filter 3 is the same as Filter 2 except the decision threshold is re-estimated one window after every beat rather than every window.

Listener 1:

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	2.5	3.0	4.0	N/A
30dB	3.5	3.5	4.5-5.0	N/A
30dB	3.5	N/A	4.5-5.0	4.0

Listener 2:

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	1.0	2.0	4.0	N/A
30dB	2.0	3.0	5.0	N/A
30dB	2.0	N/A	4.0	3.0

Listener 3:

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	1.0	2.0	4.0	N/A
30dB	3.0	2.0	4.0	N/A
30dB	3.0	N/A	4.0	3.0

Listener 4:

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	1.0	2.5	4.0	N/A
30dB	3.0	3.5-4.0	5.0	N/A
30dB	3.0	N/A	4.5-5.0	4.0

Listener 5:

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	1.5	3.0	4.0	N/A
30dB	3.0	4.0	5.0	N/A
30dB	3.0	N/A	5.0	4.0

Table 5-8: Listening test results for each listener.

Noise Level	Noisy Signal	Filter 1	Filter 2	Filter 3
20dB	1.4	2.5	4	N/A
30dB	2.9	3.2	4.7	N/A
30dB	2.9	N/A	4.4	3.6

Table 5-9: Average test results for all listeners

5.5.1 Interpretation of Listening Results

Filter 2 contains both filter improvements: harmonic prediction, and allowing the filter to pass the windows that beats occur in. Three of the four listener's tests rated Filter 2 a 5, which is CD quality for an input noise of 30 dB. For an input noise of 20 dB, the ratings dropped to 4, which is still better than the original Filter 1 without the new features. This is due to the fact that the noise becomes more audible and discernible on the beats with the 20 dB input noise level. Since the filter passes the windows with the tempo beats, you not only get the percussion event without any filtering but the noise present without any suppression as well. In the case of 30dB input noise, the percussion events are loud enough to partially mask the noise. Due both to this and the noise being present in periodic bursts corresponding to the beat, the ear has a hard time discerning the noise from the percussion event since they occur at the same time. When the noise level is increased the periodic bursts of noise become more obvious since they are not being as well masked by the simultaneous percussion event.

Filter 3 is the same as Filter 2 except the threshold is re-estimated one window after a beat rather than every window. The subjective listening tests reveal Filter 3 as being noticeably worse than Filter 2, but the same or better than Filter 1. Since the threshold and thus the filter are not optimal for each window, there is a perceived degradation in the filtered audio. However, if the user can live with the degradation in quality, a much faster filtering algorithm is achieved. The time taken to filter an audio signal with Filter 3 is considerably shorter than either of the other two filters. However computational speed was not an important consideration in determining if one filter was better than another.

5.5.2 Analysis of Other Results

In section 4.1.1, the formulation of the threshold is discussed. There is an offset denoted as lambda that raises the noise detection threshold directly by the lambda value. This offset ensures that the threshold is sitting above the level of the noise in frequency bins containing only noise. As this offset value increases, more noise is suppressed, which is desirable. However, lower level harmonics also get suppressed since they may no longer be above the new threshold raised by the increased offset. Harmonic prediction allows for the noise threshold offset to be increased above the level that works best with the original filter without the same loss in quality. This is due to the fact that the harmonic prediction algorithm passes the harmonics that would normally get suppressed by the raised threshold, because they are multiples of a lower frequency that was passed and detected by the harmonic peak detector.

Also investigated was whether setting the processing and filter re-estimating window size to match the duration of a particular length of musical note (quarter note, eighth note, sixteenth, *etc.*) resulted in any improvement in performance. It turns out that basing the window lengths on note duration has the largest impact on the performance with percussion. The windows are re-aligned so the beginning of a window corresponds to the beat onset. The optimal window length in this case would be the decay time of the percussion event. If the window length is longer than the time it takes for the percussion event to decay extra noise from the end of the decay to the beginning of the next window will be heard. If the window is shorter than the time it takes for the percussion event to decay the drum hit will be prematurely filtered and will sound clipped off. Also, the

lengths of the segments corresponding to note durations will rarely be a power of two, which is best for increasing processing speed due to using radix-2 Fast Fourier Transforms in the computations. Zero padding can be added in order to effectively use the FFT algorithm. However these factors make window length based on note duration impractical.

6 Conclusion

Filtering broadband noise from an audio signal is a unique problem since broadband noise is not confined to a narrow bandwidth. It is spread across many frequencies, including those that include the signal components that need to be preserved. This makes static, classic linear filters, *i.e.* low pass, high pass, band pass, and notch filters, essentially useless for broadband noise reduction.

The technique used in this thesis approaches the noise reduction from the harmonic side of the signal rather than the noise. This method selects frequency bins in the magnitude spectrum that clearly contain components of the original signal and leaves them unprocessed. The frequency bins that contain noise are then attenuated using a set of psycho-acoustical rules. This technique has already been developed and demonstrated for speech enhancement [1]. The goal of this thesis was to show that this basic concept could be improved upon. This study demonstrated that the use of harmonic prediction when filtering solo piano indeed provides improvement in signal to noise ratio while also decreasing distortion. Harmonic prediction also allows the offset for the signal detection threshold to be raised higher than with the original filter, allowing improved noise reduction without loss of harmonic signal content and audio fidelity. The signal to noise ratio for a drum solo is greatly improved when the filter is allowed to pass the windows that contain percussion hits. The distortion is also greatly reduced in this case.

Ultimately perception by the human ear is the most important measure of audio quality. While the quantitative signal to noise ratios are worse for the filtered excerpts of rock music than the noise corrupted input signal to noise ratios, the subjective listening

tests show that the audio excerpts filtered using the improved noise filter are perceived as being significantly improved.

7 Future Studies

1) **Improve beat tracking.**

   The beat tracking algorithm used in this thesis determines the beat locations based on a tempo that is determined for the entire song. If the tempo changes somewhere in the song, the beat spacing will change. This means the beat tracking algorithm needs to be able to track changes in tempo and adjust the beat spacing which controls which windows will be passed by the filter.

2) **Optimize constants used in formulation of threshold**

   When the adaptive threshold is being calculated, there are three constants used to scale the different parts of the threshold. There is a scale factor for the median filter, the averaging filter, and an offset. How these scale factors were chosen is not discussed in [1]. Optimizing these scale factors may possibly provide further improvement to the filter results. Developing a means to self-adjust and optimize these parameters based on the audio signal content or noise characteristics could provide additional performance benefits.

8 Bibliography

[1] Kauppinen, I.; Roth, K.; , "An adaptive psychoacoustic filter for broadband noise reduction in audio signals," *Digital Signal Processing, 2002. DSP 2002. 2002 14th International Conference on*, vol.2, no., pp. 963- 966 vol.2, 2002

[2] Painter T, Spanias A, A Review of Algorithms for Perceptual Coding of Digital Audio Signals.

[3] Wikipedia contributors. "Bark scale." *Wikipedia, The Free Encyclopedia.* Wikipedia, The Free Encyclopedia, 17 Jan. 2012. Web. 10 Feb. 2010.

[4] Boll, S.; , "Suppression of acoustic noise in speech using spectral subtraction," *Acoustics, Speech and Signal Processing, IEEE Transactions on* , vol.27, no.2, pp. 113-120, Apr 1979

[5] Ephraim, Y.; Malah, D.; , "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator," *Acoustics, Speech and Signal Processing, IEEE Transactions on* , vol.32, no.6, pp. 1109- 1121, Dec 1984

[6] Ben Aicha, S. Ben Jebara and D. Pastor "Speech denoising improvement by musical tones shape modification," International Symposium on Communication, Control and Signal Processing ISCCSP, Morocco 2006.

[7] "DOLBY B, C, AND S NOISE REDUCTION SYSTEMS: Making Cassettes Sound Better," Dolby Laboratories Inc., San Francisco, CA

[8] J. Laroche. Efficient tempo and beat tracking in audio recordings. Journal of the Audio Engineering Society, 51(4):226–233, Apr 2003.

[9] D. Ellis (2007). Beat Tracking by Dynamic Programming. *J. New Music Research*, Special Issue on Beat and Tempo Extraction, vol. 36 no. 1, March 2007, pp. 51-60. (10pp)

[10] Digital Signal Processing: Principles, Algorithms, and Applications, Proakis and Manolakis, 4th ed., Prentice Hall.

[11] Analog and Digital Signal Processing, Ambardar A., 2nd ed., Brooks/Cole.

[12] E. Zwicker and E. Terhardt, "Analytical expression for critical-band rate and critical bandwidth as a function of frequency," *J. Acoust. Soc. Am.*, vol. 68, pp. 1523-1525, (1980 Nov.).

9 Appendix A: Broadband Filter code

The first three functions listed are the filtering functions. The filtering functions require the input signal to already be digitized and stored in a variable. In Matlab the wavread function is used to input a file. The first function BBNFilt filters broadband noise with the addition of harmonic prediction. The second function BBNFiltBPHP has both harmonic prediction and beat location detection and relaxes the filter in the processing windows that contain beats. The third function BBNFiltBPHPAB is the same as the second function except the decision threshold is re-estimated one window after a window containing a beat. The three filtering functions are followed by the peak detector used in the harmonic prediction, the masking threshold calculator to determine the level of suppression for frequency bins classified as noise, and the tempo analysis code for determining beat locations.

9.1 Filter with harmonic prediction

```
function a=BBNFilt(y,offset,channel,fs,noise)

%{
Main function for filtering broadband noise.  This function reads in a
wav file and outputs the filtered signal.  Y is the input wav file,
offset is the value that goes into lambda which controls the offset of
the threshold, fs is the sample rate, and noise is the input signal to
noise ratio in dB.  This filter applies harmonic prediction. The
variable “a” which is the filtered signal is then placed in an output
variable of the users choosing.
%}

b=y(:,channel);
clear y

% segment length
L=8192;


%This option adds white noise
z=awgn(b,noise,'measured');
```

```
%Calculate number of segments
nsegs=floor((length(b))/L);

%Initialize all arrays to proper size
Mask=[];
seg=zeros(nsegs,2*L);
fftseg=zeros(nsegs,2*L);
filter_spec=zeros(nsegs,2*L);
mag_sq_spec=zeros(nsegs,2*L);
amp=zeros(nsegs,2*L);
Y2=zeros(nsegs,2*L);

a=zeros(1,length(z)+L);
i=1;

%Moving average with length of 21
avg_length=21;  % Use odd number
beta=10;  %Scale factor


%Median filter with length 201
alpha=16;  %Scale factor
med_length=201;



N=2*L; % Filter Spectrum length

%Threshold offset
lambda=offset;



% Add phase for 1/2 h[n] length delay to the filter frequency response
% h[n] length assumed to be N/2 for linear convolution so the delay is
%N/4 samples.  Without this, the filter impulse response is non-causal,
%centered on n=0; so the DFT circular convolution produces time domain
%samples at the END of the ifft sequence.

k=[-N/2:N/2-1]; % FFT indices for the window function
delayangle=(-2.*pi.*(N/4).*k./N);  % -2*pi*ndelay*k/N
delayangle=ifftshift(delayangle);  % shift angles for k=0->(N-1)

for k=1:nsegs
    disp(['Segment ', int2str(k), ' of ', int2str(nsegs)])
    peak_num=1;

    % Extract an L length segment of data, Zero pad to 2L length

    seg(k,:)=[z((k*L)-(L-1):k*L);zeros(L,1)];

    fftseg(k,:)=fft(seg(k,:));  % Data spectrum
```

```
    % Compute the magnitude squared spectrum
    % Normalize by L
    mag_sq_spec(k,:)=abs(fftseg(k,:)).^2/L;

    %Apply moving average to segment
    Avg=beta.*(conv(ones(1,avg_length)/avg_length,mag_sq_spec(k,:)));
    Avg=Avg((avg_length-1)/2+1 : 2*L+((avg_length-1)/2));

     % Median filter the spectrum with f=0 in center to eliminate edge
    % effects

r1=alpha.*ifftshift(medfilt1(fftshift(mag_sq_spec(k,:)),med_length));

     %Calculate threshold for segment
    T=r1-Avg+lambda;


      %Calculate masking threshold for frame and apply decision rule to
      %determine filter spectrum.  Update magnitude spectrum
      [Mask]=SupThresh(N,mag_sq_spec(k,:),fftseg(k,:),fs);
      Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

      filter_spec(k,:)=((mag_sq_spec(k,:)>=T)|(mag_sq_spec(k,:)<Mask))+
      sqrt(Mask).*((mag_sq_spec(k,:)<T)&(mag_sq_spec(k,:)>=Mask));
      mag_sq_spec(k,:)=mag_sq_spec(k,:).*filter_spec(k,:);

    %Second iteration of masking threshold
      [Mask]=SupThresh(N,mag_sq_spec(k,:),fftseg(k,:),fs);
      Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

      filter_spec(k,:)=((mag_sq_spec(k,:)>=T)|(mag_sq_spec(k,:)<Mask))+
      sqrt(Mask).*((mag_sq_spec(k,:)<T)&(mag_sq_spec(k,:)>=Mask));

    %Determine peak locations
    [peaks]=pkdet(mag_sq_spec(k,1:8192),0.8).*(fs./N);



    freq_index_conversion=N./fs;
    passband_width = 20;

    %Use to limit upper harmonic frequency
    harmonic_cutoff=3000;

   %Place pass bands at harmonics of peaks detected by the peak
detector
    if size(peaks,1)<8
      while peak_num<=size(peaks,1)
        for harmonic_count = 1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
```

```
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
                filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;
            end
        end
        peak_num=peak_num+1
      end


    else
      while peak_num<9
        for harmonic_count=1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
           elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
           filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

             end
        end
        peak_num=peak_num+1;
      end
    end


   %Smooth the filter response by convolving with a Hamming window
   r_smooth=conv(hamming(5)./sum(hamming(5)), filter_spec(k,:));
```

```
   filter_spec(k,:)=r_smooth(3:length(r_smooth)-2);



 % Make the filter real by imposing even symmetry on magnitude and odd
 % symmetry on phase
 filter_spec(k,:)=[filter_spec(k,1:size(filter_spec,2)/2+1),
conj(filter_spec(k,size(filter_spec,2)/2:-1:2))];

    % Spectral multiplication
    Y2(k,:)=fftseg(k,:).*filter_spec(k,:).*exp(j.*delayangle);

   seg_out(k,:)=real(ifft(Y2(k,:)));

    %Overlap and add segments to reconstruct song
    a(i:i+2*L-1)=a(i:i+2*L-1)+seg_out(k,:);
    i=i+L;


end

%Choose appropriate "middle" samples of reconstructed song
a=a(1+(N/4):length(a)-(N/4));
```

9.2 Filter with harmonic prediction and passing windows that contain beats

```
function a=BBNFiltBPHP(y,offset,channel,fs,noise)
%{
This function filters broadband noise while determining and allowing
the filter to pass the beat locations.  Harmonic prediction is also
integrated into this filtering function.  Y is the input wav file,
offset is the value that goes into lambda which controls the offset of
the decision threshold, fs is the sample rate, and noise is the input
SNR in dB.  The variable "a" is the filtered song.
%}

b=y(:,channel);
clear y

%Segment length
L=4096;

%This corrupts the input with AWGN with a SNR specified by "noise"
z=awgn(b,noise,'measured');

%This function determine the beat locations and returns them to the
%variable "beats"
```

```
[beats]=mytempo(z,fs);
beats=round(beats);

%Initialize all arrays to proper size
seg=zeros(1,2*L);
fftseg=zeros(1,2*L);
mag_sq_spec=zeros(1,2*L);
first_half=zeros(1,2*L);
fftfirst_half=zeros(1,2*L);
filter_spec=zeros(1,2*L);
seg_out=zeros(1,2*L);
Y2=zeros(1,2*L);
Mask=[];

%Moving average with length of 21
avg_length=21;  % Use odd number
beta=10;

%Median filter with length 201
alpha=16;
med_length=201;

%Threshold offset value
lambda=offset;

%Filter spectrum length
N=2*L;

%Calculate phase shift so that filter impulse response is causal
k=[-N/2:N/2-1]; % FFT indices for the window function
delayangle=(-2.*pi.*(N/4).*k./N);  % -2*pi*ndelay*k/N
delayangle=ifftshift(delayangle);  % shift angles for k=0->(N-1)

%Initialize output array
a=zeros(1,length(b)+L);

%Initialize counter and flag variables
i=1;
k=1;
pos=1;
flag=0;
flag2=0;
beat_num=1;
peak_num=1;


while pos<length(b)-L

disp(round(k))
    if pos==beats(beat_num)
        flag=1;
    end

    %Extract L length segment and zero pad to N length
```

```
    seg=[z(pos:pos+L-1)' zeros(1,L)];
    pos=pos+L;

    %Data spectrum
    fftseg=fft(seg);

    %Magnitude squared spectrum normalized by length L
    mag_sq_spec=abs(fftseg).^2/L;

    %Apply moving average to segment
    Avg=beta.*(conv(ones(1,avg_length)/avg_length,mag_sq_spec));
    Avg=Avg((avg_length-1)/2+1 : 2*L+((avg_length-1)/2));

    %Apply median filter to spectrum with f=0 in center to eliminate
edge
    %effect
    r1=alpha.*ifftshift(medfilt1(fftshift(mag_sq_spec),med_length));

    %Calculate threshold for segment
    T=r1-Avg+lambda;

    %Check if beat_num is less than the size of the beats array
    if beat_num<size(beats,2)

        %Check if a beat happens in the current segment.  If true then
the current segment is split into two segments.  The first being from
the start of the segment to the beat location and the second from the
beat location to length L.  The first part of the segment is filtered
using the filter from one segment before.  The second part is passed by
he filter as well as the beat location being the new start point for
the next segment. The beginning of each segment is re-aligned with the
beat locations.
        if (beats(beat_num)>=pos+1-L) & (beats(beat_num)<=pos)
            firstwin=round(beats(beat_num)-(pos-L));
            num_zeros=round(N-firstwin);
            first_half=[seg(1:firstwin) zeros(1,num_zeros)];
            fftfirst_half=fft(first_half);

            fftseg=fftfirst_half;
            pos=beats(beat_num);
            flag2=1;

        else

%Calculate masking threshold for frame apply decision rule to determine
filter spectrum.  Update magnitude squared spectrum.
                [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
                Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));
                mag_sq_spec=mag_sq_spec.*filter_spec;

                %Second iteration of masking threshold.
                [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
```

```
                Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));


                %Determine peak locations.
                [peaks]=pkdet(mag_sq_spec(1:8192),0.068).*(fs./N);



    freq_index_conversion=N./fs;
    passband_width = 20;

    %Use to limit upper harmonic frequency
    harmonic_cutoff=5000;

 %Place pass bands at harmonics of peaks detected by the peak detector
    if size(peaks,1)<8
      while peak_num<=size(peaks,1)
        for harmonic_count = 1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
                filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;
            end
        end
        peak_num=peak_num+1;
      end


    else
      while peak_num<9
        for harmonic_count=1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(1:min([round((peaks(peak_num).*harmonic_count +
```

```
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
             filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

              end
          end
          peak_num=peak_num+1;
       end
    end

%Smooth the filter response by convolving with a Hamming window
                r_smooth=conv(hamming(5)./sum(hamming(5)),
filter_spec);
                filter_spec=r_smooth(3:length(r_smooth)-2);

                %Make filter response real by imposing even symmetry on
                %magnitude and odd symmetry on phase.
                filter_spec=[filter_spec(1:size(filter_spec,2)/2+1),
conj(filter_spec(size(filter_spec,2)/2:-1:2))];

        end
    else

%If no beat occurs in window then filter normally.  Calculate masking
threshold and apply decision rule to segment.  Update magnitude squared
spectrum.
            [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
            Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));
            mag_sq_spec=mag_sq_spec.*filter_spec;

            %Second iteration of masking threshold
            [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
            Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));

            %Determine peak locations in magnitude squared spectrum
            [peaks]=pkdet(mag_sq_spec(1:8192),0.068).*(fs./N);
```

```
    freq_index_conversion=N./fs;
    passband_width = 20;

    %Use to limit upper harmonic frequency
    harmonic_cutoff=5000;

    %Place pass bands around harmonics of peaks found by peak detector
    if size(peaks,1)<8
      while peak_num<=size(peaks,1)
        for harmonic_count = 1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
                filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;
            end
        end
        peak_num=peak_num+1;
      end


    else
      while peak_num<9
        for harmonic_count=1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
           elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
```

```
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
             filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

              end
        end
        peak_num=peak_num+1;
      end
    end

%Smooth the filter response by convolving with a Hamming window
            r_smooth=conv(hamming(5)./sum(hamming(5)), filter_spec);
            filter_spec=r_smooth(3:length(r_smooth)-2);

%Make filter real by imposing even symmetry on magnitude and odd
symmetry on phase
            filter_spec=[filter_spec(1:size(filter_spec,2)/2+1),
conj(filter_spec(size(filter_spec,2)/2:-1:2))];

    end




%Filter re-aligned segment which starts on the beat
    if flag==1
        Y2=fftseg.*exp(j.*delayangle);
        i=beats(beat_num);
        beat_num=beat_num+1;
        flag=0;

%Filter first part of segment containing a beat using the filter from
    %previous segment
    elseif flag2==1
        Y2=fftseg.*filter_spec.*exp(j.*delayangle);
        flag2=0;

%Filter segment containing no beat
    else
        Y2=fftseg.*filter_spec.*exp(j.*delayangle);

    end
    seg_out=real(ifft(Y2));

%Overlap and add segments to reconstruct song
    a(i:i+2*L-1)=a(i:i+2*L-1)+seg_out;
    i=i+L;
    k=k+1;
```

```
end

%Choose appropriate "middle" segments of reconstructed song
a=a((L/2)+1:length(a)-(L/2));
```

9.3 Filter with threshold re-estimated one window after beat, with harmonic prediction, and passing windows that contain beats

```
function a=BBNFiltBPHPAB(y,offset,channel,fs,noise)

%This function filters broadband noise while determining beat locations
and allowing the filter to pass the segments that contain beats.
Harmonic prediction is also integrated into this filter.  This filter
also re-estimates the threshold one segment after a beat rather than on
every segment.  Y is the input wav file, offset if controls the
variable lambda which is the offset of the decision threshold, channel
is which channel to filter (1 for left and 2 for right), fs is the
sample rate, and noise is the SNR of the AWGN added to the signal in
dB.  Variable "a" is the filtered output.

b=y(:,channel);
clear y

%Segment length
L=4096;

%This corrupts the input with AWGN with an SNR specified by "noise"
z=awgn(b,noise,'measured');

%This function determines the beats locations
[beats]=mytempo(z,fs);
beats=round(beats);

%Initialize all arrays to proper size
seg=zeros(1,2*L);
fftseg=zeros(1,2*L);
mag_sq_spec=zeros(1,2*L);
first_half=zeros(1,2*L);
fftfirst_half=zeros(1,2*L);
filter_spec=zeros(1,2*L);
seg_out=zeros(1,2*L);
Y2=zeros(1,2*L);
Mask=[];
```

```
%Moving average with length of 21
avg_length=21;  % Use odd number
beta=10;

%Median filter with length 201
alpha=16;
med_length=201;

%Threshold offset value
lambda=offset;

%Filter spectrum length
N=2*L;

k=[-N/2:N/2-1]; % FFT indices for the window function
delayangle=(-2.*pi.*(N/4).*k./N);  % -2*pi*ndelay*k/N
delayangle=ifftshift(delayangle);  % shift angles for k=0->(N-1)

%Output array
a=zeros(1,length(b)+L);

%Initialize counters and flags
i=1;
k=1;
pos=1;
flag=0;
flag2=0;
beat_num=1;
peak_num=1;


while pos<length(b)-L

disp(round(k))
    if pos==beats(beat_num)
        flag=1;
    end


    %Extract L length segment and zero pad to N length
    seg=[z(pos:pos+L-1)' zeros(1,L)];
    pos=pos+L;

    %Data spectrum
    fftseg=fft(seg);

    %Check is beat_num is less than the size of the beats array
    if beat_num<size(beats,2)

Check if a beat happens in the current segment.  If true then the
current segment is split into two segments.  The first being from the
start of the segment to the beat location and the second from the beat
location to length L.  The first part of the segment is filtered using
the filter from one segment before.  The second part is passed by the
```

```
filter as well as the beat location being the new start point for the
next segment. The beginning of each segment is re-aligned with the beat
locations.
        if (beats(beat_num)>=pos+1-L) & (beats(beat_num)<=pos)
            firstwin=round(beats(beat_num)-(pos-L));
            num_zeros=round(N-firstwin);
            first_half=[seg(1:firstwin) zeros(1,num_zeros)];
            fftfirst_half=fft(first_half);

            fftseg=fftfirst_half;
            pos=beats(beat_num);
            flag2=1;

        %Check is the position either the first segment or one segment
after a beats has occurred.  If this is true re-estimate decision
threshold.
        elseif (pos==L+1)|(pos==beats(beat_num)+2*L)
                mag_sq_spec=abs(fftseg).^2/L;

Avg=beta.*(conv(ones(1,avg_length)/avg_length,mag_sq_spec));
                Avg=Avg((avg_length-1)/2+1 : 2*L+((avg_length-1)/2));

r1=alpha.*ifftshift(medfilt1(fftshift(mag_sq_spec),med_length));
                T=r1-Avg+lambda;

                %Calculate masking threshold for current frame and
apply decision rule to determine filter spectrum.  Update magnitude
squared spectrum.
                [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
                Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));
                mag_sq_spec=mag_sq_spec.*filter_spec;

                %Second iteration of masking threshold.
                [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
                Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));



                %Determine peak locations is magnitude squared
spectrum.
                [peaks]=pkdet(mag_sq_spec(1:8192),0.068).*(fs./N);



    freq_index_conversion=N./fs;
    passband_width = 20;

    %Use to limit upper harmonic frequency
    harmonic_cutoff=5000;
```

```
    %Place pass bands at harmonics of peaks detected by the peak
detector.
    if size(peaks,1)<8
      while peak_num<=size(peaks,1)
        for harmonic_count = 1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
                filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;
            end
        end
        peak_num=peak_num+1;
      end


    else
      while peak_num<9
        for harmonic_count=1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
           elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
          filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
            filter_spec(round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

             end
```

```
        end
        peak_num=peak_num+1;
      end
    end
                %Smooth the filter response by convolving with a
Hamming window
                r_smooth=conv(hamming(5)./sum(hamming(5)),
filter_spec);
                filter_spec=r_smooth(3:length(r_smooth)-2);

                %Make the filter real by imposing even symmetry on
                %magnitude and odd symmetry on phase
                filter_spec=[filter_spec(1:size(filter_spec,2)/2+1),
conj(filter_spec(size(filter_spec,2)/2:-1:2))];
                beat_num=beat_num+1;
        else
                %If not first segment or segment after a beat
filter_spec not re-estimated.
                filter_spec=filter_spec;
        end
    %Check is the position is either the first segment or one segment
after a beats has occurred.  If this is true re-estimate decision
threshold.
    elseif (pos==L+1)|(pos==beats(beat_num)+2*L)
            mag_sq_spec=abs(fftseg).^2/L;

Avg=beta.*(conv(ones(1,avg_length)/avg_length,mag_sq_spec));
            Avg=Avg((avg_length-1)/2+1 : 2*L+((avg_length-1)/2));

r1=alpha.*ifftshift(medfilt1(fftshift(mag_sq_spec),med_length));
            T=r1-Avg+lambda;

            %Calculate masking threshold for current frame and apply
decision rule to determine filter spectrum.  Update magnitude squared
spectrum.
            [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
            Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));
            mag_sq_spec=mag_sq_spec.*filter_spec;

            %Second iteration of masking threshold.
            [Mask]=SupThresh(N,mag_sq_spec,fftseg,fs);
            Mask=[Mask(1:size(Mask,2)) Mask(size(Mask,2):-1:1)];

filter_spec=((mag_sq_spec>=T)|(mag_sq_spec<Mask))+(sqrt(Mask).*((mag_sq
_spec<T)&(mag_sq_spec>=Mask)));

            %Determine peak locations in magnitude squared spectrum.
            [peaks]=pkdet(mag_sq_spec(1:8192),0.068).*(fs./N);



    freq_index_conversion=N./fs;
```

```
    passband_width = 20;

    %Use to limit upper harmonic frequency
    harmonic_cutoff=5000;

    %Place pass bands around harmonics of peaks detected by peak
detector.
    if size(peaks,1)<8
      while peak_num<=size(peaks,1)
        for harmonic_count = 1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
            elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
                filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;
            end
        end
        peak_num=peak_num+1;
      end


    else
      while peak_num<9
        for harmonic_count=1:20
            if (((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion))<0.5

filter_spec(k,1:min([round((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

                %Use to limit upper harmonic frequency
           elseif (((peaks(peak_num).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion))>((N./fs).*harmonic_cutof
f);
           filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((harmonic_cuto
ff)*freq_index_conversion),size(filter_spec,2)]))= 1.0;
                else
            filter_spec(k,round((peaks(peak_num).*harmonic_count -
(0.5.*passband_width))*freq_index_conversion):min([round((peaks(peak_nu
```

```
m).*harmonic_count +
(0.5.*passband_width))*freq_index_conversion),size(filter_spec,2)]))=
1.0;

             end
        end
        peak_num=peak_num+1;
      end
    end

            %Smooth the filter frequency response by convolving with a
            %Hamming window.
            r_smooth=conv(hamming(5)./sum(hamming(5)), filter_spec);
            filter_spec=r_smooth(3:length(r_smooth)-2);

            %Make filter real by imposing even symmetry on magnitude
and odd symmetry on phase
            filter_spec=[filter_spec(1:size(filter_spec,2)/2+1),
conj(filter_spec(size(filter_spec,2)/2:-1:2))];
    else
            %If not first segment or segment after a beat filter_spec
not re-estimated.
            filter_spec=filter_spec;
    end

   %Filter re-aligned segment which starts on the beat
    if flag==1
        Y2=fftseg.*exp(j.*delayangle);
        i=beats(beat_num);
        flag=0;



    %Filter first part of segment containing a beat using the filter
    %calculated for the previous segment
    elseif flag2==1
        Y2=fftseg.*filter_spec.*exp(j.*delayangle);
        flag2=0;

    %Filter segment containing no beat.
    else
        Y2=fftseg.*filter_spec.*exp(j.*delayangle);
    end
    seg_out=real(ifft(Y2));

    %Overlap and add segments to reconstruct song
    a(i:i+2*L-1)=a(i:i+2*L-1)+seg_out;
    i=i+L;
    k=k+1;


end

%Choose appropriate "middle" segments of reconstructed song
a=a((L/2)+1:length(a)-(L/2));
```

9.4 Peak Detection code

```
function [maxindex]=pkdet(v, delta)

%This function takes the magnitude squared spectrum of the corrupted
%spectrum along with a constant as the inputs and outputs the peak
%locations in samples.  The constant delta is a tolerance.  If the
%difference between a peak and its surrounding samples is at least
%delta then that peak is actually a peak.
maxindex = [];
minindex = [];
v = v(:);
x = (1:length(v))';

min = Inf; max = -Inf;
minpos = NaN; maxpos = NaN;

findmax = 1;

for i=1:length(v)
    this = v(i);
    if this > max, max = this; maxpos = x(i); end
    if this < min, min = this; minpos = x(i); end

    if findmax
        if this < max-delta
            maxindex = [maxindex ; maxpos];
            min = this; minpos = x(i);
            findmax = 0;
        end
    else
        if this > min+delta
            minindex = [minindex ; minpos];
            max = this; maxpos = x(i);
            findmax = 1;
        end
    end
end
```

9.5 Code for masking threshold

```
function Mask=SupThresh(window,mag_sq_spec,fftseg,fs);

%This function calculates the masking threshold for each segment.
Window is the filter spectrum length, mag_sq_spec is the magnitude
squared spectrum for the current segment, fftseg is the data spectrum
for the current segment, and fs is the sample rate.
```

```
Mask=[];
vf=zeros(1,window/2);   %vector to hold freq. values
vz=zeros(1,window/2);   %vector to hold Bark values

%Total number of critical bands
ZT=ceil(13*atan(0.00076*(fs/2))+3.5*atan(((fs/2)/7500)^2));

%Vector to hold number of points per critical band
N=zeros(1,ZT);

%Initialize vector to hold sample numbers corresponding to edges of
each critical band
critlimit=zeros(2,ZT);
critlimit(1,1)=1;
critlimit(2,ZT)=window/2;




%convert fft indexes to frequency and barks
for ii=1:window/2
    f=(fs*(ii-1))/length(fftseg);
    vf(ii)=f;
    z=13*atan(0.00076*f)+3.5*atan((f/7500)^2);
    if z==0

z=13*atan(0.76*(0.5*fs/window)/1000)+3.5*atan(((0.5*fs/window)/7500)^2)
;
    end
    vz(ii)=z;
    N(ceil(z))=N(ceil(z))+1;
    if ii>1 & floor(z)-floor(vz(ii-1))>0
        critlimit(2,ceil(z)-1)=ii-1;
        critlimit(1,ceil(z))=ii;
    end
end

%Calculate the Energy per critical band
E=zeros(1,ZT);
for ii=1:ZT
    E(ii)=sum(mag_sq_spec(critlimit(1,ii):critlimit(2,ii)));
end

%Calculate the approximation of the basilar membrane spreading function
%using normalized Bark(z) scale
B=zeros(1,ZT);
Bdb=zeros(1,ZT);
temp=floor(ZT/2);
for ii=1:ZT
    z=(ii-temp);
    Bdb(ii)=15.91+7.5*(z+0.474)-17.5*sqrt(1+(z+0.474)^2);
end
```

```
%Convert B from dB's
B=10.^(Bdb/10);

%Calculate the spread masking across critical bands
C=conv(E,B);
temp1=round(length(E)/2);
C=C(temp1:ZT+temp1-1);  %Grab correct "middle" samples

%Geometric mean of Energy
Gm=prod(E)^(1/ZT);

%Arithmetic mean of Energy
Am=sum(abs(E))*(1/ZT);

%Calculate spectral flatness measure
SFM=10*log10(Gm/Am);
SFMmax=-60;

%Generate index of tonality
%delta=min(SFM/SFMmax,1);
delta=1;



%calculate the offset for the masking threshold
for ii=1:ZT
    O(ii)=delta*(14.5+ii)+(1-delta)*5.5;
end

Mraw=10.^(log10(C)-(O/10)); %Raw masking threshold
M=Mraw./N; %Final normalized threshold

%Convert threshold back to Hz frequency

for ii=1:ZT
    Mask=[Mask repmat(M(ii),1,N(ii))];
end
```

9.6 Code for beat location

```
function [beats]=mytempo(d,sr)

%This function takes the corrupted wav file specified by “d” and the
sample rate “sr” as its inputs and outputs the beat locations in
samples.

z=d;

%Segment length corresponding to 10msec
L=441;
```

```
%Initialize arrays to proper size
r=zeros(floor(length(z)/L),L);
G=zeros(floor(length(z)/L),L);
a=zeros((floor(length(z)/L)-1),L);
r1=zeros(floor(length(z)/L),L);


%Loop that calculates magnitude spectrum for 10msec segments and takes
the  square root of each segment and stores it in G

    for k=1:floor(length(z)/L)

        r(k,:)=z((k*L)-(L-1):k*L);
        r1(k,:)=fft(r(k,:));
        r1(k,:)=abs(r1(k,:));
        G(k,:)=r1(k,:).^(1/2);

    end


%Loop computes differences in G and sums up values that are between
100Hz and 10kHz

i=1;
    for n=1:(floor(length(z)/L)-1)

        a(i,:)=G(n+1,:)-G(n,:);
        b(i)=sum(a(i,1:100));
        i=i+1;
    end


%Half wave rectification of b.
b=b.*(b>0);
clear n


%Generate onset strength envelopes corresponding to tempos of 60 up to
150BPM in 1BPM increments using impulses on the beat locations.  Cross
correlate each tempos onset strength envelope with the one calculated
for the song.  The largest correlation corresponds to the tempo of the
song.
for n=60:150;

bpm=n;     %Beats per minute
bps=bpm/60;%Convert to beats per second
window=.01;
fs=44100;   %Sample rate
W=6000;
WPB=1/(bps*window);
ER(n,1:WPB:W)=1;

corr(n)=max(xcorr(b,ER(n,:)));
```

```
end

%Using the onset strength enveloped calculated for the song and the
determined tempo period calculate the beat locations that best fit the
onset strength envelope.
2007-06-19 Dan Ellis dpwe@ee.columbia.edu
alpha=200;
BPM=find(corr==max(corr));

%Onset envelope sample rate
oesr=fs/L;

%Beat period in samples
period=(60*oesr)/BPM;

localscore=b;
backlink=-ones(1,length(localscore));
cumscore=localscore;
prange=round(-2*period):-round(period/2);
txwt=(-alpha*abs((log(prange/-period)).^2));


for i=max(-prange+1):length(localscore)

    timerange=i+prange;
    scorecands=txwt+cumscore(timerange);
    [vv,xx]=max(scorecands);
    cumscore(i)=vv+localscore(i);
    backlink(i)=timerange(xx);
end

[vv,beats]=max(cumscore);

while backlink(beats(1))>0
    beats=[backlink(beats(1)),beats];
end
beats=beats./100;
beats=beats.*sr;
```